UNIVERSIDADE ESTADUAL DA PARAÍBA
CAMPUS I - CAMPINA GRANDE
CENTRO DE CIÊNCIA E TECNOLOGIA
DEPARTAMENTO DE COMPUTAÇÃO
CURSO DE CIÊNCIA DA COMPUTAÇÃO

JONATHAN SANTOS PINTO

UMA PROPOSTA DE SISTEMA DE VOTAÇÃO BASEADO EM BLOCKCHAIN

CAMPINA GRANDE
2022

JONATHAN SANTOS PINTO

UMA PROPOSTA DE SISTEMA DE VOTAÇÃO BASEADO EM BLOCKCHAIN

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da Computação da Universidade Estadual da Paraíba, como requisito parcial à obtenção do título de bacharel em Ciência da Computação.

Área de concentração: Sistema Distribuído.

Orientadora: Profª. Drª. Kézia de Vasconcelos Oliveira.

CAMPINA GRANDE
2022

P659p Pinto, Jonathan Santos.
Uma proposta de sistema de votação baseado em Blockchain [manuscrito] / Jonathan Santos Pinto. - 2022.
79 p. : il. colorido.

Digitado.
Trabalho de Conclusão de Curso (Graduação em Computação) - Universidade Estadual da Paraíba, Centro de Ciências e Tecnologia , 2022.
"Orientação : Profa. Dra. Kézia de Vasconcelos Oliveira , Coordenação do Curso de Computação - CCT."

1. Blockchain. 2. Segurança de dados. 3. Votação eletrônica. 4. Protocolo de confiança. I. Título

21. ed. CDD 005.1

Elaborada por Camile de A. Gomes - CRB - 15/559 BC/UEPB

JONATHAN SANTOS PINTO

UMA PROPOSTA DE SISTEMA DE VOTAÇÃO BASEADO EM BLOCKCHAIN

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da Computação da Universidade Estadual da Paraíba, como requisito parcial à obtenção do título de bacharel em Ciência da Computação.

Área de concentração: Sistema Distribuído.

Aprovada em 03 de Agosto de 2022.

BANCA EXAMINADORA

Kézia de Vasconcelos Oliveira Dantas

Profa. Dra.Kézia de Vasconcelos Oliveira Dantas (DC - UEPB)
Orientador(a)

Sabrina de F. Souto.

Profa. Dra. Sabrina de Figueiredo Souto (DC - UEPB)
Examinador(a)

Paulo Bh

Prof. Dr. Paulo Eduardo e Silva Barbosa (DC - UEPB)
Examinador(a)

DEDICATÓRIA

A minha família, a qual dedico todo o meu esforço.

AGRADECIMENTOS

Agradeço a Deus pelo dom da vida!

Agradeço a Professora Drª Kezia Vasconcelos, por toda orientação e compreensão na condução deste trabalho.

Agradeço aos membros da banca por toda disponibilidade e participação na avaliação desta pesquisa.

Agradeço aos meus pais por toda batalha que enfrentaram para me garantir a melhor educação possível.

Agradeço à minha esposa Renata e ao meu filho John por fazerem parte da minha vida e das minhas conquistas.

Agradeço aos meus amigos da Universidade que caminharam comigo por toda essa jornada, em especial Abimael, que batalhou bastante ao meu lado.

Por fim, gostaria de agradecer a Universidade Estadual da Paraíba pelo apoio, presteza e dedicação em oferecer um Curso Público de Computação de qualidade ao qual estou tendo a honra de me formar.

RESUMO

Os sistemas de votação sempre foram questionados, seja votação em papel ou eletrônica/digital. O formato de apuração dos votos, transmissão e divulgação dos resultados, para diversas pessoas é uma verdadeira caixa preta. O avanço tecnológico permitiu que os processos se tornassem mais democráticos, seguros e transparentes. Considerando esse contexto, justifica-se a importância de encontrar mecanismos que aumentem a confiabilidade dos sistemas de votação, dado isto, o objetivo deste trabalho é apresentar uma solução que garanta confiabilidade e segurança através das características de segurança de um sistema de Blockchain. Para tanto, objetivamente se faz necessário conceituar o que é Blockchain e suas características, demonstrar casos de uso que comprovem a adotem da respectiva tecnologia em áreas diversas e implementar um sistema que utilize destas especificações para um sistema de votação genérico, definindo métricas para avaliação dos dados coletados. Dentre as contribuições desta pesquisa, pode-se destacar a utilização de uma Blockchain para registrar os votos de um sistema de votação de forma segura e imutável, com um desempenho aderente a grandes quantidades de votos.

Palavras-chave: Blockchain. Segurança. Confiabilidade. Voto.

ABSTRACT

Voting systems have always been questioned, whether paper voting or electronic/digital . The format for counting votes, transmission and dissemination of results, for many people is a real black box. Technological advances have allowed processes to become more democratic, secure and transparent. Considering this context, the importance of finding mechanisms that increase the reliability of voting systems is justified, in order to this, the objective of this work is to present a solution that guarantees reliability and security through the immutability characteristics of a Blockchain system. Therefore, it is objectively necessary to conceptualize what Blockchain is and its characteristics, demonstrate use cases that prove the adoption of the respective technology in different areas and implement a system that uses these specifications for a generic voting system, defining metrics for evaluating the collected data. Among the contributions of this research, we can highlight the use of a Blockchain to register the votes of a voting system in a secure and immutable way, with a performance regarding the large amounts of votes.

Keywords: Blockchain. Safety. Reliability. Vote.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API	Interface de Programação de Aplicação
ASIC	Circuitos Integrados de Aplicação Específica
AWS	Amazon Web Services
BaaS	Blockchain as a Service (Blockchain como serviço)
CPU	Unidade Central de Processamento (Central Processing Unit)
DAO	Data Access Object
DApp	Aplicativo Descentralizado
DPoS	Delegated Proof of Stake (Prova de Participação por Delegação)
ETH	Ether
GBC	Conselho Global de Blockchain
GPU	Unidade de Processamento Gráfico
IDE	Ambiente de Desenvolvimento Integrado
JDK	Kit Java de Desenvolvimento
JS	JavaScript
MVC	Model-View-Controller
NARP	Agência Nacional de Registros Públicos
NIST	Instituto Nacional de Padrões e Tecnologia
NSA	Agência de Segurança Nacional dos Estados Unidos
ONU	Organização das Nações Unidas
P2P	Rede Ponto a Ponto
PBFT	Tolerância a falhas bizantinas práticas
PoET	Proof of Elapsed Time (Prova de Tempo Decorrido)
PoS	Proof of Stake (Prova de Participação)
PoW	Proof of Work (Prova de Trabalho)
QLDB	Amazon Quantum Ledger Database
RF	Requisitos Funcionais
RNDS	Rede Nacional de Dados em Saúde
RNF	Requisitos Não Funcionais
SDK	Kit de desenvolvimento de software
SHA	Secure Hash Algorithms

SQL	Structured Query Language
SUS	Sistema Único de Saúde
UML	Linguagem de modelagem unificada
WFP	World Food Programme (Programa Mundial de Alimentos)

SUMÁRIO

1 INTRODUÇÃO

Segundo Silveira (2011), no Brasil, desde 1932, vem-se buscando transformar o sistema de eleições com o objetivo de democratizar o voto e evitar fraudes eleitorais, no entanto, segundo o Tribunal Superior Eleitoral, somente em 1996 é datada a versão oficial da urna eletrônica brasileira. Diversos questionamentos surgiram desde que este novo modelo de votação eletrônico foi implantado, tais como: A urna eletrônica realmente é segura? Uma vez registrado o voto, este voto pode ser alterado? É possível inserir novos votos na contagem final da votação?

O *Jornal El País*, publicou em 2018 uma notícia debatendo sobre questionamentos que o atual presidente do Brasil e o ex-presidente dos Estados Unidos da América expressaram colocando em xeque seus respectivos sistemas eleitorais, devido à divulgação de hipóteses de violação dos votos registrados.

Estes questionamentos não se restringem somente ao sistema eleitoral governamental, mas em todo sistema de votação, seja de cargos políticos ou de reitorias de Universidades. Revoredo (2018) afirma que governos e instituições buscam frequentemente tecnologias que aumentem o nível de confiabilidade, transparência e segurança dos sistemas de votação. A mesma propõe que um sistema de *Blockchain* reduzirá potenciais fraudes e erros na contagem de votos (principalmente para sistemas que ainda utilizam a votação em papel).

O ritmo acelerado do avanço da tecnologia busca, entre outros aspectos, garantir segurança no tratamento e armazenamento das informações, notadamente quando se trata de dados financeiros e bancários, Aparecida (2014). A WEB 3.0 impulsionada por conceitos como Computação de Borda, Descentralização, Inteligência Artificial, Aprendizado de Máquina e *Blockchain*, vai além de conectar serviços, mas busca também conectar indivíduos, corporações e máquinas suficientemente inteligentes de forma segura, Ejeke (2022).

Entre estes novos conceitos, tem-se destacado o *Blockchain* devido sua ascensão perante aos sistema de criptomoedas e principalmente o Bitcoin, afirma Sales (2021). Grandes players do mercado tecnológico como Facebook, Google, Amazon e IBM, vêm investindo elevadas quantidades de dinheiro no estudo e implementação de Blockchains voltadas aos negócios, Revista Forbes (2022).

O Blockchain não se limita somente a sistemas financeiros, benefícios como descentralização, transparência e segurança são características de uma Blockchain

e estão sendo explorados em outros campos, como medicina, mídia, governo, indústrias e entre outros, Bashir (2020).

Bashir (2020) define Blockchain como um livro-razão distribuído em uma rede ponto a ponto que é criptograficamente seguro, imutável e atualizável somente através de um algoritmo de consenso acordado entre pares na rede.

Diante desta perspectiva, percebe-se a necessidade de compreender e implementar um sistema de votação online utilizando-se de *Blockchain* para registro de votos como solução para as questões de segurança e segurança.

Portanto, indaga-se: sistemas *Blockchain* realmente garantem segurança e segurança nos votos registrados em um sistema de votação?

O objetivo geral deste trabalho é implementar uma solução para um sistema de votação generalizado, utilizando de uma Blockchain permissionada, contrato inteligente e características comuns a uma Blockchain: confiabilidade e segurança.

Para tanto, foram delineados os seguintes objetivos específicos: apresentar e conceituar o que é *Blockchain,* demonstrando através de exemplos de casos reais sua aplicação nas diversas áreas; apresentar uma solução para um sistema de votação baseado em *Blockchain* e implementar uma interface web para registros de votos; identificar e apresentar métricas para verificação dos dados coletados.

Parte-se da hipótese de que os sistemas atuais de votação são questionáveis, e um novo sistema se faz necessário, pois levantou-se a hipótese de falhas a caráter da coleta e armazenamento dos votos registrados nos sistemas atuais.

Assim, para viabilizar o teste da hipótese, realiza-se uma pesquisa empírica, com procedimentos bibliográficos e documentais.

Na seção 2.1 a 2.10, são descritos os conceitos de Blockchain, características e funcionamento segundo a literatura.

Na seção 2.11 apresentaremos alguns casos de uso reais de Blockchain nas mais diversas áreas de negócios e sistemas humanitários.

Na seção 3 definiremos a metodologia utilizada para desenvolver a solução proposta, como análise de requisitos, ferramentas utilizadas, padrão do projeto, funcionalidades, desenvolvimento da aplicação, implementação, execução e teste da solução.

Por fim, na seção 4 iremos apresentar os resultados da pesquisa, concluindo na seção 5 que os objetivos são atendidos e a pesquisa respondida, indicando que

se faz necessário uma evolução dos sistemas atuais de votação, buscando uma nova estratégia para a tentativa de equacionar o problema de confiança do sistema eleitoral.

2 REFERENCIAL TEÓRICO

Nesta seção iremos contextualizar e apresentar os principais conceitos de uma Blockchain, principais características, quais elementos fazem parte de um bloco, conceitos de redes peer-to-peer, criptografia, os tipos de blockchain e os principais mecanismos de consenso.

2.1 O que é um Blockchain

A palavra Blockchain vem de:

Block = bloco + chain = cadeia

Ou seja, blocos em cadeia, ou blocos encadeados. Revoredo (2019) diz que a palavra Blockchain foi nomeada pelo mercado para definir um conjunto de várias tecnologias programáveis, com o objetivo de registrar, verificar e rastrear qualquer coisa com valor.

O Blockchain tinha como objetivo ser um ledger (livro-razão) público, onde todas as transações de todos os usuários do Bitcoin fossem armazenadas de forma que o processo não gerasse um gasto duplo, ou seja, a mesma moeda não fosse negociada duas vezes e que não exigisse a necessidade de uma entidade central processando tal negociação. Logo ficou claro que tal aplicação também poderia ser utilizada em outras áreas onde existisse a necessidade de registrar informações desde sua criação até as últimas modificações, afirma Unias (2016).

Embora bastante conhecida e ligada às criptomoedas, a Blockchain pode ser utilizada em diversos segmentos: contratos, eleições, ajuda humanitária, registro de terras, gerenciamento de energia, cadeia de suprimentos, sistemas financeiros, combate a fake news e entre outros. Veremos alguns destes casos adiante.

2.2 Como funciona a Blockchain

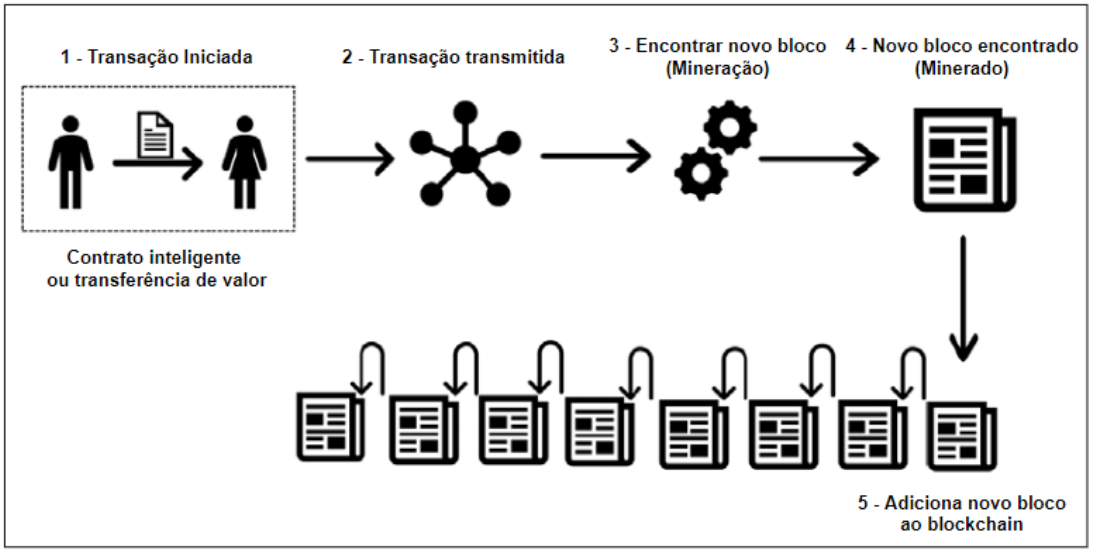
De um modo geral o Blockchain funciona como um banco de dados distribuído, onde as transações são realizadas entre nós (rede peer-to-peer), e as informações da transação são armazenadas de forma encadeada. Todos os membros da rede têm acesso ao registro das informações, e tudo isto sem a

necessidade de um agente mediador centralizado, tome como exemplo de agente mediador um banco, afirma Unias (2019).

A Blockchain armazena as informações das transações em blocos e todos os envolvidos possuem uma cópia idêntica dos registros das transações. As informações uma vez registradas não podem ser alteradas ou apagadas. As transações são confirmadas através de um mecanismo chamado “consenso” que normalmente é obtido através de algoritmos matemáticos complexos. Os blocos sequenciais formam uma cronologia de todos os acontecimentos, e desta forma é possível encontrar o bloco gênese, como é chamado o primeiro bloco da cadeia de blocos. Durante a transação, os participantes da rede validam a transação, garantindo confiança e transparência, diz Revoredo (2019).

Bashir (2020) definiu a estrutura básica do funcionamento de uma Blockchain, na figura 1 veremos como uma Blockchain valida as transações e cria novos blocos:

Figura 1 - Geração de um novo bloco na Blockchain



Fonte: Imagem adaptada do livro Mastering Blockchain (2020).

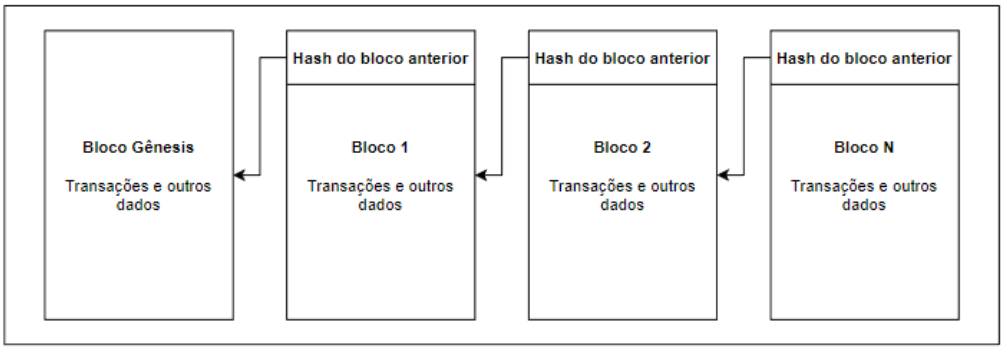
1. Um nó (usuário da Blockchain) inicia uma nova transação, assinando-a digitalmente com sua chave privada, que é única e pessoal. Uma transação na Blockchain pode representar diversas ações (veremos algumas na sessão de casos de usos da Blockchain). As regras de negócios são aplicadas à transação, envolvendo a ação e o endereço de destino.

2. A transação inicial é validada e propagada na rede através de protocolos de envio de dados, para outros nós validarem a transação conforme mecanismos definidos para a Blockchain (definiremos os mecanismos de validação mais adiante).
3. Nesta etapa inicia-se o processo de mineração, onde os nós mineradores buscam um novo bloco.
4. Após a resolução do quebra-cabeça matemático, ou resolução do mecanismo de consenso, o bloco é considerado criado ou encontrado. E desta forma a transação é considerada confirmada.
5. O novo bloco é validado pelos pares do nó, e as transações ou contratos inteligentes são executados e propagados na rede. Neste novo bloco uma ligação é definida com o bloco anterior (chamado de ponteiro hash).

2.3 Elementos genéricos de um Blockchain

Abaixo demonstramos uma estrutura genérica de um Blockchain conforme a figura 2:

Figura 2: Estrutura genérica de uma Blockchain

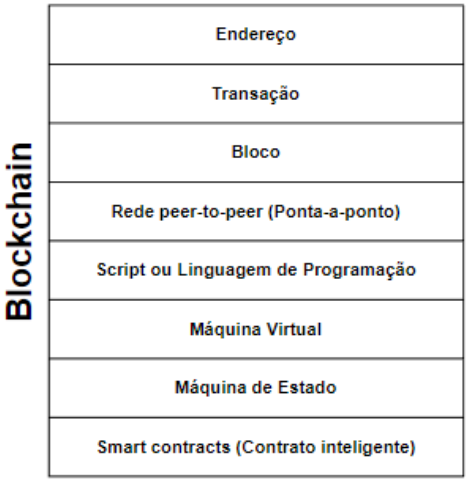


Fonte: Elaborada pelo autor (2022).

Conforme já comentamos, a Blockchain é formada por uma cadeia de blocos, onde cada bloco possui informações de transações realizadas, permitindo assim rastreabilidade de todas as transações realizadas. Cada bloco possui um apontador para o bloco anterior, até que se chegue ao bloco Genesis. O bloco Gênesis é o primeiro bloco no Blockchain, aponta Bashir (2020).

Bashir (2020) no livro Mastering Blockchain, define a Blockchain nas seguintes partes de um todo, conforme figura 3:

Figura 3: Componentes de uma Blockchain



Fonte: Imagem adaptada do livro Mastering Blockchain (2020).

Comentaremos abaixo cada uma das partes definidas na imagem acima.

2.3.1 Endereço

O endereço é uma representação exclusiva utilizada para cada remetente e destinatário na Blockchain. Geralmente é uma chave pública. Quando falamos de endereço no mundo de criptomoedas, este será o local ao qual os cripto ativos estão associados.

2.3.2 Transação

A transação é a unidade que movimenta a Blockchain. A transação representa a transferência de informações de um endereço para outro. No universo das criptomoedas, uma transação representa um valor monetário, como por exemplo Bitcoin ou Ethereum. Em outras situações, a transação pode ser de informações referentes à execução de programa de um contrato inteligente.

Normalmente os nós envolvidos nas transações são representados por chaves públicas, chave esta gerada no cadastro de entrada da Blockchain. As

chaves públicas também representam o endereço digital de uma carteira de criptomoedas.

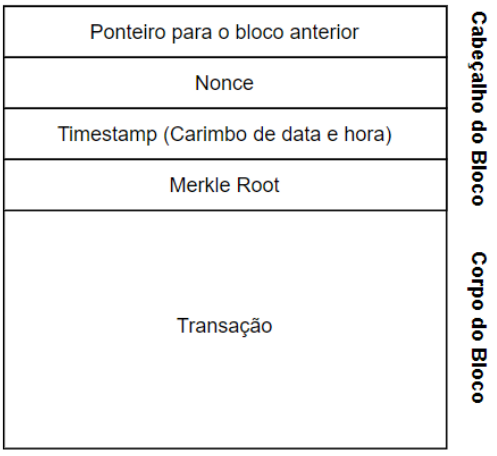
Hyperledger, (2018) diz que as estruturas de negócio na Blockchain consistem em duas etapas principais:

1. Ordenação das transações - Nesta etapa é recebida a transação solicitada pelo cliente. A ordenação pode ser implementada de diversas maneiras e o conteúdo da transação é protegido por hash ou criptografia.
2. Validação das transações - Esta camada depende da lógica de negócios aplicada à Blockchain através dos contratos inteligentes. É nesta camada que a transação é validada, garantido as regras do contrato inteligente. Por outro lado, as transações inválidas são rejeitadas e descartadas.

2.3.3 Bloco

O bloco é composto por um cabeçalho e o conteúdo da transação. Podemos visualizar a estrutura básica de um bloco na Blockchain na figura 4:

Figura 4: Estrutura genérica de um bloco



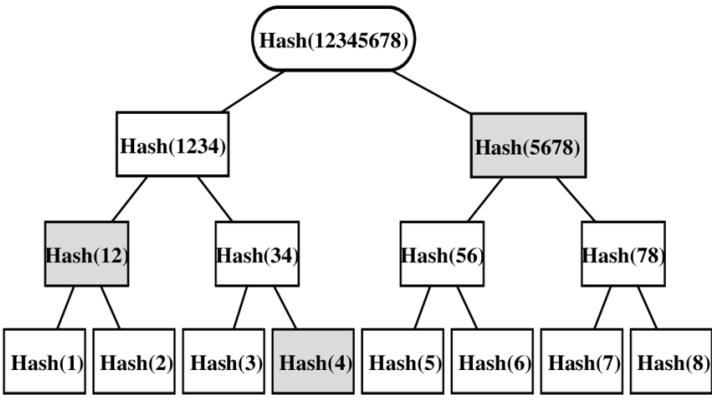
Fonte: Imagem adaptada do livro Mastering Blockchain (2020).

Cabeçalho do bloco:

- **Ponteiro para o bloco anterior -** Esta referência é o hash do bloco anterior. (Veremos sobre hash mais a frente).

- **Nonce -** É um número gerado e utilizado apenas uma vez. É um número aleatório destinado a autenticação e transferência de dados entre duas partes.
- **Timestamp** - É um carimbo contendo a hora e a data de criação do bloco.
- **Merkle Root -** É o hash de todos os hashes combinados das transações no bloco. Esse mecanismo permite uma verificação rápida de dados do Blockchain, bem como o movimento rápido de grandes quantidades de dados de um nó de um computador para outro na rede Blockchain, afirma Frankenfield (2021), na figura 5 podemos visualizar a ilustração de uma árvore merkle:

Figura 5: Exemplo de uma Árvore Merkle



Fonte: SeekPNG

- **Corpo do bloco:**

As transações contemplam o corpo do bloco. A transação nada mais é do que um registro de um evento. O corpo do bloco varia de acordo com o design da Blockchain, Bashir (2020).

2.3.4 Rede Peer-to-peer

Em sua publicação *"Bitcoin: A Peer-to-Peer Electronic Cash System"* Satoshi Nakamoto tinha como principal objetivo resolver o problema de confiança entre partes, e esta ruptura somente foi possível através da arquitetura peer-to-peer do projeto Bitcoin. O conceito Peer-to-peer (ou rede ponto a ponto) vem da ideia da

interação entre duas partes (conhecidas ou desconhecidas), onde os nós são arranjados em uma rede sem depender de uma entidade mediadora, ou seja, não há necessidade de um servidor central.

As cargas das transações são distribuídas por toda a rede. Revoredo (2019) explica que este mecanismo possibilitou na rede bitcoin a criação de uma camada de incentivo econômico para aqueles participassem da organização da rede, Satoshi Nakamoto nomeou este mecanismo de Proof of Work ou Prova de Trabalho (veremos mais adiante de forma de detalhada o conceito de Proof of Work).

Os computadores da rede peer-to-peer executam um software e armazenam uma cópia atualizada dos registros das transações.

> [...]permitindo transferência de valor entre pessoas que não se conhecem ou não confiam entre si, sem a necessidade de um intermediário, através de consenso obtido por algoritmos. Nele, as transações são armazenadas de forma imutável e transparente para todos. Revoredo (2019).

2.3.5 Script ou Linguagem de Programação

Os scripts ou programas de uma Blockchain são comandos responsáveis pela execução das operações de transação de um endereço para outro. A linguagem geralmente é limitada e não permite desenvolvimento arbitrário de programas, afirma Bashir (2020).

2.3.6 Máquina de Estado

Como resultado da execução das transações, a Blockchain pode ser vista como um mecanismo de máquina de estado, ao qual modifica seus estados a cada interação na rede.

2.3.7 Smart Contracts (Contrato Inteligente)

Contratos Inteligentes são programas com pequenos códigos encapsulados, automatizados e autônomos executados em cima do Blockchain, contendo todas as regras de negócios e condições a serem atendidas pelos seus usuários, emulando a

lógica das cláusulas de um contrato real. Bashir (2020) diz que este recurso oferece flexibilidade e segurança nos cenários que são aplicados.

Os contratos inteligentes estão diretamente ligados com a questão principal da Blockchain: realizar transações com entidades desconhecidas, as quais não temos confiança. Comprovando esta afirmação, Revoredo (2009) diz que os contratos inteligentes formalizam relações comerciais na internet, inteiramente peer-to-peer, sem a necessidade de uma entidade intermediadora.

Pelo fato de a Blockchain possuir uma linguagem limitada (não-Turing), ou seja, somente aceita entradas determinadas, os contratos inteligentes também são limitados a operações aritméticas, lógicas e criptografias básicas. Uma evolução referente aos contratos inteligentes foi realizada na Blockchain Ethereum, ao qual implementou uma linguagem completa de programação (Turing-completa), permitindo contratos inteligentes mais sofisticados.

> Esses programas de computador são executados por várias partes na rede Ethereum e, portanto, têm a capacidade de operar de forma autônoma e independente do controle de qualquer indivíduo. [...] Eles permitem a rastreabilidade do desempenho de contrato em tempo real e, portanto, podem economizar custos, proporcionando mais transparência, responsabilidade e menos burocracia, eis que a conformidade e o controle acontecem na hora. (Revoredo (2019) apud Buterin, 2014).

Além de transações econômicas, os contratos inteligentes podem ser utilizados para executar e verificar qualquer tipo de propriedade de terra, controle de acesso ou propriedades de direitos autorais.

> Contratos inteligentes na Web 3.0 permitem que as empresas se envolvam diretamente com seus consumidores sem a participação de terceiros. Empresas podem desenvolver contratos inteligentes e acompanhar seu sucesso usando dados. Não haverá perda ou roubo de dados devido à sua natureza descentralizada. (Ejeke, 2022),

2.4 Criptografia

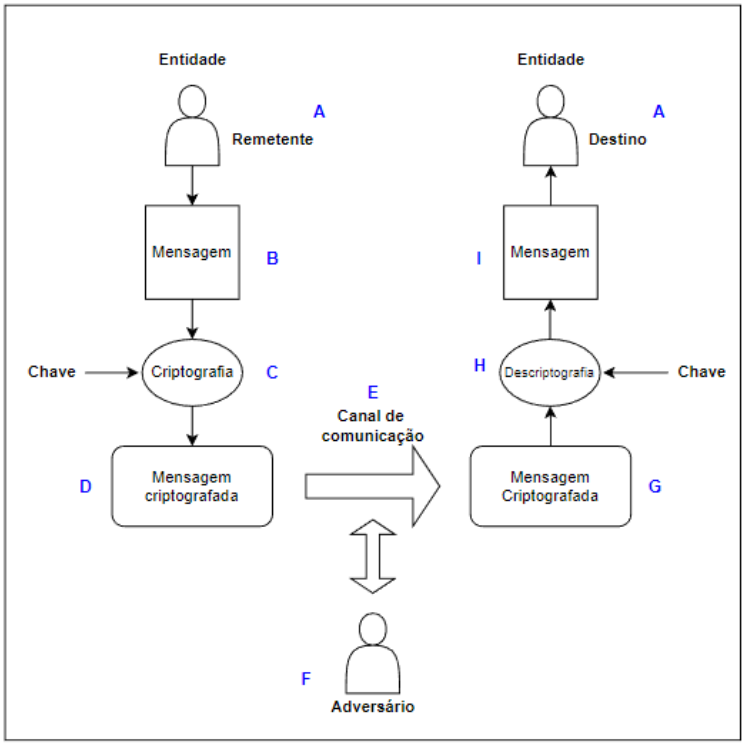
Bashir (2020) diz que Criptografia é a ciência de tornar a informação segura na presença de adversários. O mecanismo de criptografia e descriptografia requer uma chave secreta para organizar os dados de forma legível, desta forma, mesmo

que os dados sejam interceptados, os mesmos não poderão ser acessados, pois as informações estarão embaralhadas e incompreensíveis.

Além de chaves criptografadas, o Blockchain utiliza outras camadas de segurança: funções hash, assinatura digital e criptografia de chave pública. Tais serviços oferecem além de segurança, integridade e autenticação de entidade, afirma Bashir (2020).

Na figura 6 podemos visualizar um modelo criptográfico genérico:

Figura 6: Modelo de criptografia genérica



Fonte: Imagem adaptada do livro Mastering Blockchain (2020).

Na imagem anterior podemos identificar as entidades do sistema (**A**), que podem ser remetentes e destinatários (assim como em um sistema de correspondências). Após o envio da mensagem (**B**), o mecanismo de criptografia (**C**) transforma a mensagem em um texto embaralhado (**D**). Abaixo segue exemplo de um texto criptografado (formando um hash) através de alguns algoritmos de criptografia:

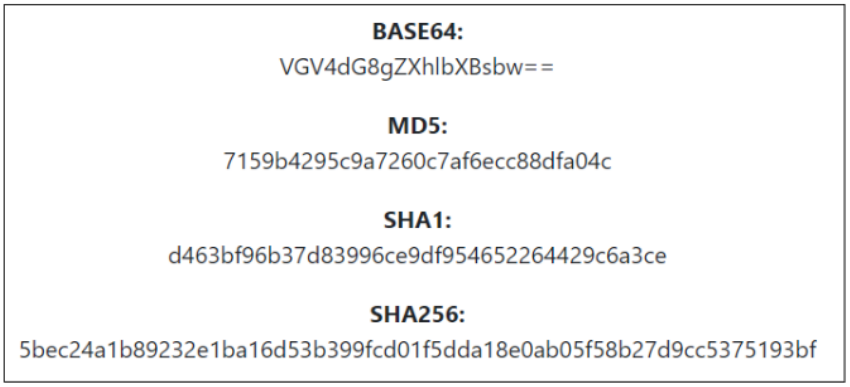
Na figura 7 podemos ver a mensagem original:

Figura 7: Texto original antes de ser criptografado

Fonte: Ferraz (2022)

Na figura 8 podemos ver a mensagem criptografada:

Figura 8: Mensagem criptografada em vários algoritmos



Fonte: Ferraz (2022)

Após criptografar os dados, a mensagem é enviada através de um canal de comunicação (**E**). No entanto, durante a comunicação os dados podem ser interceptados por um atacante que chamamos de adversário (**F**). Entenda o adversário como qualquer entidade que tente burlar o serviço de segurança. Após receber a mensagem (**G**), a mesma é descriptografada (**H**) e a mensagem é entregue desembaralhada (ou descriptografada) para o destinatário (**I**).

Vale ressaltar que alguns algoritmos não permitem o processo de descriptografia do hash gerado, como é o exemplo do SHA1 e do SHA256, este processo é chamado de criptografia definitiva, diz Ferraz (2021).

2.5 Funções Hash

As funções hash são utilizadas para criar resumos de comprimento fixo (no caso do SHA256 tem tamanho de 64 caracteres) independente do tamanho da entrada. As funções que geram o hash normalmente são iteradas e dedicadas.

Existem várias famílias de funções hash, como: MD, SHA1, SHA2, SHA-3 entre outras. As funções hash também são utilizadas para fornecer serviços de integridade aos dados, afirma Bashir (2020).

Em particular, a Blockchain do Bitcoin utiliza o SHA-256 duas vezes para verificar o esforço computacional gasto pelos mineradores. O SHA-256 foi desenvolvido pela Agência de Segurança Nacional dos Estados Unidos (NSA) e o Instituto Nacional de Padrões e Tecnologia (NIST). O objetivo do mecanismo é gerar hashes com base em padrões aos quais protejam os documentos/informações de qualquer agente externo que tente modificá-los, continua Bashir (2020).

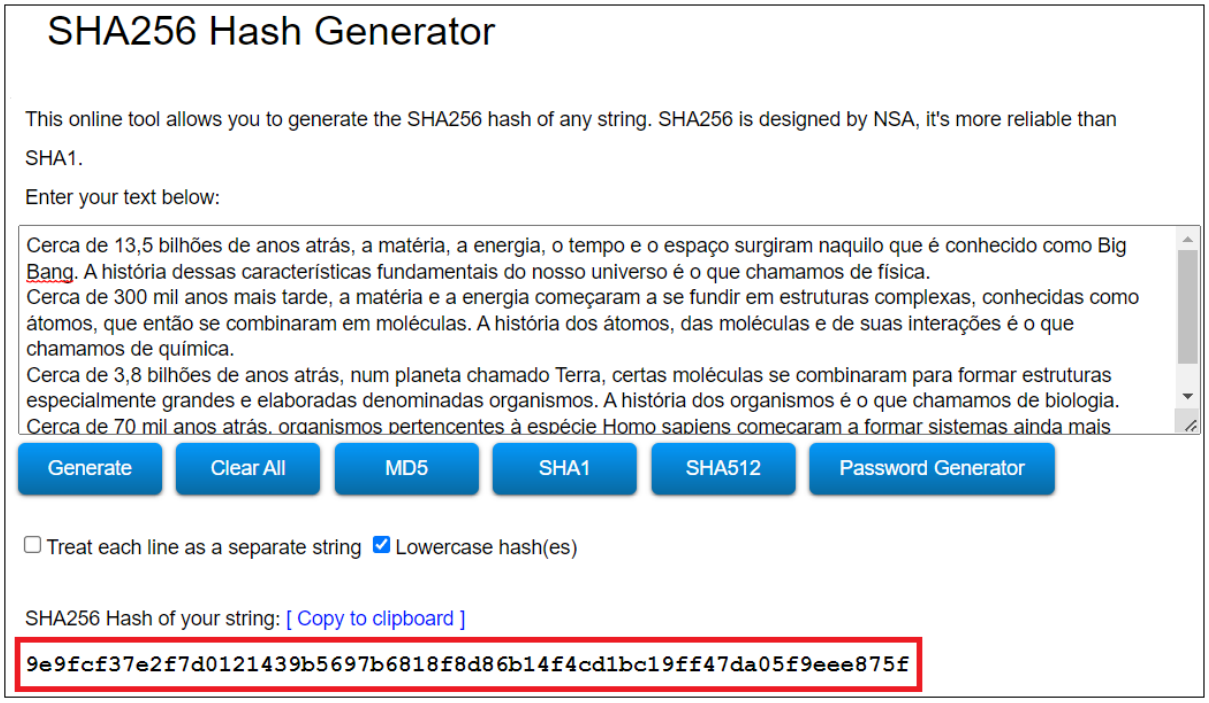
> Entre as maneiras diferentes de criar hashes, o algoritmo usado pelo SHA-256 é um dos mais utilizados para o equilíbrio entre segurança e custo computacional de geração, pois é um algoritmo muito eficiente para a alta resistência à colisão que possui. (Bit2me Academy, 2022).

Uma peculiaridade do SHA-256 é que a sequência de letras e números sempre terá como resultado 64 caracteres, independente do tamanho do seu conteúdo. Por exemplo:

Todas as palavras contidas no livro “Sapiens - Uma breve história da humanidade”, assim como uma única letra do alfabeto estariam expressas em um hash de 64 caracteres.

Na figura 10 podemos visualizar em destaque de vermelho, um exemplo de Hash gerado pelo algoritmo SHA-256 contendo quinze linhas de um trecho do livro “Sapiens”. Gerador de Hash SHA-256 disponível no site: https://passwordsgenerator.net/sha256-hash-generator/

Figura 9: Hash gerado pelo algoritmo SHA256



Fonte: Passwords Generator

2.6 Características de um Blockchain

Revoredo (2019) destaca algumas características potenciais que uma Blockchain deve proporcionar aos seus participantes:

- **Descentralização:** quando tratamos de redes públicas (qualquer pessoa pode entrar ou sair da Blockchain a qualquer momento), a ideia é que nenhum órgão ou entidade tenha autoridade sobre a rede. Para que novas decisões sejam tomadas deve haver um consenso majoritário de 50% + 1 na rede (veremos mais a frente sobre questões de tomadas de decisões em uma Blockchain).
- **Privacidade**: Esta é uma das principais características que torna a Blockchain relevante para o mercado em geral, pois garante aos envolvidos total ou parcial anonimato (pode variar em algumas Blockchains), utilizando chaves públicas e funções hash (código alfanumérico) como proteção das informações. A privacidade na Blockchain é garantida através da troca segura de informações protegidas por criptografia. Tal mecanismo é possível através das seguintes tecnologias:

- **Transparência** - O incentivo na confiança entre os envolvidos é o ponto chave do design da Blockchain, desta forma os usuários devem ter acesso livre às informações, com exceção das chaves privadas citadas anteriormente. Já nas Blockchains privadas (ou permissionadas) os gerentes da rede decidem o grau de transparência que os registros devem ter.

2.7 Tipos de Blockchain

Laurence (2017) categoriza a Blockchain em três tipos:

- Blockchain Pública (não-permissionada)
- Blockchain Privada (permissionada)
- Blockchain Híbrida

A **Blockchain Pública** como o próprio nome já diz, todos podem participar e entrar e sair da rede no momento que desejarem. No entanto, embora seja pública, dados pessoais devem ser informados no cadastro para garantir que o usuário é real e garantirá que irá agir com responsabilidade na rede. Na Blockchain pública os responsáveis pela manutenção da rede são os próprios usuários, e por este fato todos têm acesso ao código-fonte do projeto.

As mudanças/atualizações na rede, conforme comentado anteriormente, são de responsabilidade dos usuários, mediante consenso da maioria dos envolvidos. O Bitcoin e diversas outras criptomoedas são exemplos de utilização de uma Blockchain pública. Pelo fato de os usuários não se conhecerem, ou seja, não confiarem uns nos outros, o processo de aceitação e transação dos registros na rede é mais lento, além de não ser recomendados para empresas que querem aplicar a tecnologia internamente, pois os registros são inteiramente compartilhados com a rede, e isto poderia afetar as regras de privacidade da mesma, afirma Hoinaski (2021).

Por outro lado, a **Blockchain Privada ou Permissionada**, funciona em um ecossistema fechado e propicia papéis de controle para os indivíduos da rede, onde alguns possuem funções de administrador e/ou mantenedor, enquanto outros possuem permissão apenas para realizar transações na rede. O código-fonte pode ou não ser disponibilizado abertamente. Este é o ambiente mais utilizado nas

empresas, pois garante controle sobre os participantes e orquestra as regras de negócio. Enquanto nas Blockchain públicas a prova de veracidade de trabalho é realizada por quebra-cabeças matemáticos envolvendo vários nós da rede, na Blockchain privada os modelos de aprovação são escolhidos pelos governantes da empresa (ou consórcio de várias empresas), ressalta Revoredo (2019).

Existe ainda o modelo de **Blockchain Híbrido**, ao qual fornece os benefícios da Blockchain pública e privada. *“Uma Blockchain híbrida permite maior flexibilidade e controle sobre quais dados são mantidos em sigilo e quais são compartilhados em um ledger público, bem como oferece transações mais velozes, recursos de segurança e auditabilidade.” (Revoredo, 2019).*

2.8 Mecanismos de Consenso

Laurence (2017) em uma breve introdução aos mecanismos de consenso de um Blockchain, afirma que o poder e confiança da tecnologia Blockchain está diretamente ligada aos mecanismos de consenso aplicados à rede de nós. Tal mecanismo cria um sistema honesto sem a necessidade de uma entidade centralizada.

O mecanismo de consenso nada mais é que um acordo entre as partes envolvidas e quase sempre desconhecidas. Atualmente existem vários mecanismos de consenso disponíveis nas Blockchains, e cada um com sua finalidade. Bashir (2020) completa: *“Um mecanismo de consenso é um conjunto de etapas que são tomadas pela maioria ou por todos os nós em uma Blockchain para concordar com um estado ou valor proposto.”*

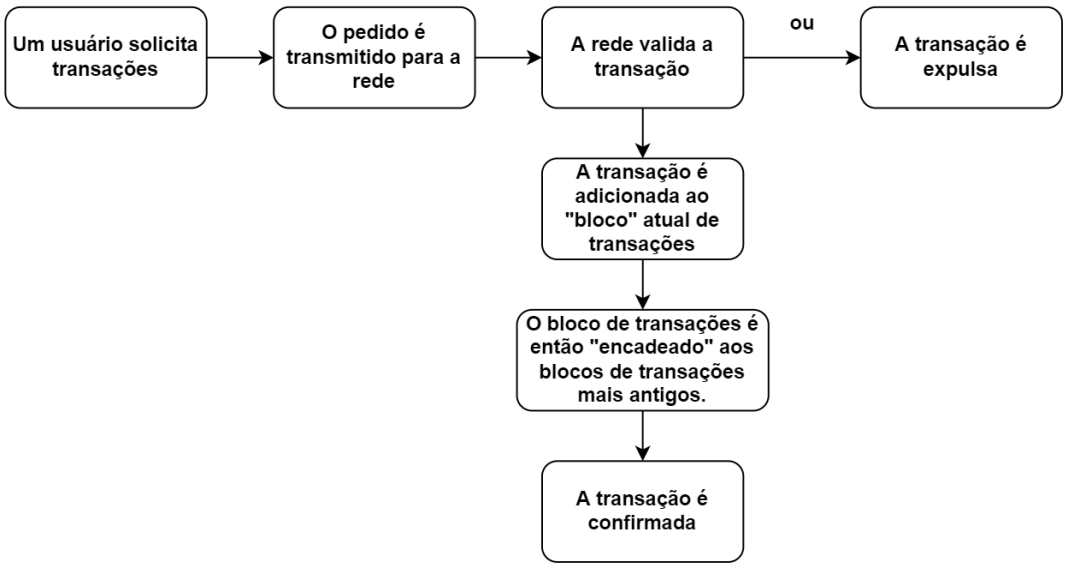
> O consenso é alcançado quando dispositivos suficientes estão de acordo sobre o que é verdadeiro e o que deve ser gravado em uma Blockchain. Portanto, os protocolos de consenso são as regras que permitem que dispositivos espalhados pelo mundo cheguem a um acordo, permitindo que uma rede Blockchain funcione sem ser corrompida. (Revoredo, 2018).

Os algoritmos de consenso também protegem a rede de ataques hackers, pois para que um ataque hacker tenha sucesso, 51% da rede precisaria ser dominada em tempo hábil, antes que a própria rede atualizasse os blocos suspeitos, e tal ação é praticamente impossível, afirma Satoshi Nakamoto (2008), pois a própria rede ao identificar uma vulnerabilidade de modificação em um bloco, enviará

atualizações para o nó desatualizado com a nova versão dos registros da Blockchain, ou seja, é quase nula a possibilidade de violação de um registro na Blockchain.

A figura 10 demonstra como a Blockchain chega a um acordo através do consenso:

Figura 10: Demonstração de um acordo via consenso



Fonte: Blockchain For Dummies, Laurence (2017).

Satoshi Nakamoto (2008) em sua publicação propôs um sistema de verificação de transações chamado de Proof-of-Work, ou “Prova de Trabalho” traduzindo para português, ou na literatura também citado como Consenso de Nakamato. O Proof-of-work é um dos mecanismos de consenso mais populares e mais utilizados em Blockchain, ressalta Xu, Weber e Staples (2019) em *Architecture for Blockchain Applications.*

Xu, Weber e Staples (2019) completam dizendo que a escolha de um determinado algoritmo de consenso afetará diretamente a segurança e escalabilidade da Blockchain. E normalmente os Blockchains utilizam apenas um mecanismo de consenso fixo, saindo da curva de padronização o *Hyperledger Fabric da IBM*, que se trata de um framework que permite vários mecanismos de consenso em uma única Blockchain.

Veremos a seguir os dois mecanismos de consenso mais populares:

2.9.1 Prova De Trabalho - Proof of Work

O modelo proposto por Satoshi Nakamoto (2008) é o mecanismo de consenso mais popular na utilização das Blockchains até os dias de hoje. O referido mecanismo tinha como principal desafio a ser resolvido *o problema dos generais bizantinos* (publicado por E. A. Akkoyunlu, K. Ekanadham, e R. V. Huber em 1975 em "Some Constraints and Trade-offs in the Design of Network Communications"). Este problema possui o seguinte questionamento: Como saber se a informação que estou recebendo/olhando não foi alterada de alguma forma, internamente ou externamente?

O problema dos generais bizantinos aborda as falhas de comunicações possível em uma rede ponto-a-ponto, ao qual no envio de uma mensagem de um ponto A até um ponto B, vários problemas poderiam acontecer e a mensagem ser interceptada e alterada, desta forma questionando a confiabilidade da informação. Este problema é um grande desafio para a ciência da computação, diz Laurence (2017) e foi resolvido através da Prova de Trabalho proposta por Nakamoto.

No tocante ao Blockchain do Bitcoin e Ethereum, os blocos são gerados através da prova de trabalho (Proof-of-work ou PoW). O PoW utiliza uma criptografia quebra-cabeça que é fácil de ser verificada e difícil de ser resolvida. Aqueles que conseguem resolver o quebra-cabeça, como prova do esforço computacional despendido, têm o direito de escrever o próximo bloco e será recompensado por tal resolução. Tal verificação demanda bastante energia elétrica dos equipamentos (este processo é conhecido como mineração). Os mineradores competem computacionalmente para resolver o quebra-cabeça, desta forma são incentivados com tokens de bitcoin para que mantenham seus equipamentos (ASICs - Circuitos Integrados de Aplicação Específica ou GPUs - Unidades de Processamento Gráfico) resolvendo os problemas criptográficos na rede em busca de gerar novos blocos, explica Xu, Weber e Staples (2019).

Resumindo: o Proof-of-work (ou prova de trabalho) é uma corrida para descobrir a combinação certa de um quebra-cabeça, aquele que descobrir é recompensando com tokens de moedas digitais e escrevem um novo bloco na Blockchain.

2.9.2 Prova de Participação - Proof of Stake

Diferente do Proof of Work, no mecanismo de consenso Proof of Stake não há recompensa por novos blocos gerados, mas sim por verificação de transações realizadas na Blockchain. Para participar do processo de validação de um bloco, os participantes devem vincular ativos, comprando bilhetes como se fosse uma “aposta”. Diante desta “aposta”, o validador do bloco é selecionado através de um sistema de “loteria” (processo de escolha por aleatoriedade entre os participantes) e um novo bloco será criado. Quanto maior a quantidade de ativos que o validador possuir, maior a participação na criação de novos blocos, afirma Singhal, Dhameja e Sekhar (2018).

> Se um minerador possui 2% de todo o Ether (ETH) na rede Ethereum, eles seriam capazes de minerar 2% de todas as transações em todo o Ethereum. Assim, quem cria o novo bloco de transação varia de acordo com o algoritmo PoS que a Blockchain está usando. (Exemplo comentado no livro “*A Beginner's Guide to Building Blockchain Solutions”. (2018). Tradução própria.)*

Este processo de aposta também funciona como uma proteção para a rede, pois para dominar a rede o atacante precisaria possuir a maior parte dos ativos, e caso a rede detectasse alguma fraude, seus ativos estariam comprometidos.

Uma vantagem do mecanismo Prova de Participação em relação à Prova de trabalho, é que não exige muito processamento, pois não existe um algoritmo de força bruta tentando organizar um quebra-cabeça, e isto resulta em economia com energia elétrica. Este mecanismo também é conhecido como consenso ecológico, afirma Revoredo (2019). Em relação à proteção contra ataques, o mecanismo punirá através da “aposta” caso identifique que o apostador é um nó malicioso, o valor apostado é retido.

2.10 Hyperledger

O Hyperledger é um projeto de código aberto e empresarial organizado pela Fundação Linux, voltado à implantação segura e desenvolvimento de padrões e estruturas para Blockchain. O hyperledger em si é constituído por grandes empresas como Accenture, Cisco, IBM, Intel e programadores entusiastas da tecnologia.

Assim como outros projetos de código aberto, é possível acessar seu código através do GITHUB: https://github.com/hyperledger.

Um dos projetos mais conhecidos do Hyperledger é o *Fabric,* que se trata de um construtor de aplicativos plug-and-play (ligar e utilizar) e é mantido pela IBM. Seu contexto é permissionado, ou seja, é necessário permissão para conseguir utilizá-lo.

> O Fabric é uma solução empresarial interessada em escalabilidade e em estar em conformidade com as normas. Todos os participantes devem registrar comprovante de identidade para serviços de adesão, a fim de adquirir acesso ao sistema. (LAURENCE, 2017)

Outro projeto importante do Projeto Hyperledger é o Sawtooth, mantido pela Intel e executado em cluster como Docker, se trata de uma plataforma onde é possível construir e executar um livro-razão distribuído para empresas. É possível acessar o site do projeto no link à seguir: https://www.hyperledger.org/use/sawtooth

Segundo o whitepaper (Hyperledger, 2018), o Sawtooth implementa alguns recursos tecnológicos ainda não comuns em Blockchains:

- Consenso dinâmico - Permite ao fornecedor da Blockchain alterar em tempo de compilação o algoritmo de consenso da Blockchain, simplesmente através de uma transação.
- Prova de tempo decorrido (PoET) - Um algoritmo de consenso voltado a escalabilidade sem um alto consumo de energia.
- Famílias de transações - Uma abstração dos contratos inteligentes, ao qual os desenvolvedores podem escrever a lógica do contrato utilizando várias linguagens de programação.
- Execução de transações paralelas - A maioria das Blockchains limitam a execução das transações de forma serializada para controlar e manter um fluxo. O Sawtooth permite a execução de transações em paralelo, dividindo em blocos os fluxos das transações. O desempenho da Blockchain é uma vantagem desse recurso de paralelismo.
- Transações privadas - O sawtooth permite a implantação de clusters de nós privados, permitindo confidencialidade através de uma cadeia distinta.

2.11 Casos de Uso

A seguir iremos listar alguns casos de uso de Blockchain, seja privada ou pública, que embora ainda seja uma tecnologia nova, está em contínuo desenvolvimento e trazendo resultados positivos em vários segmentos.

2.11.1 Ajuda humanitária

A *ONU (Organização das Nações Unidas)* através do projeto *UN WORLD FOOD PROGRAMME (Programa Mundial de Alimentos da ONU - www.wfp.org),* tem como objetivo arredacadar recursos financeiros de todo o mundo para coordenar ações de ajuda humanitária internacional. Um dos problemas enfrentados é: como indivíduos que não possuem contas bancárias ou documentos governamentais podem participar de um sistema financeiro?

A solução foi desenvolver uma plataforma que permitisse aos doadores adquirirem tokens digitais, aos quais são enviados para as famílias dos refugiados, onde os mesmos podem ser trocados por alimentos e suprimentos em redes autorizadas. A ONU criou o programa Building Blocks sobre a Blockchain Ethereum, tal plataforma contorna os problemas de depósitos e cobranças bancárias, assegurando ainda as transações com segurança. Tecnicamente o dinheiro não passa pela Blockchain, mas representa a transferência de riqueza entre as partes. Na figura 13 visualizamos o primeiro teste bem-sucedido de Building Blocks em nível de campo realizado em janeiro de 2017 no coração da província de Sindh, Paquistão.

Figura 11: Projeto Building Blocks

Fonte: WFP/Houman Haddad (2017).

> O WFP desenvolveu um aplicativo robusto na rede Building Blocks que permite rastrear, coordenar e fornecer vários tipos de assistência, incluindo dinheiro, alimentos, WASH (água, saneamento e higiene), medicamentos e muito mais. A infraestrutura técnica de Blockchain para operar a rede é baseada em um software de código aberto e é acessível gratuitamente às organizações participantes. Os aplicativos implantados na rede Building Blocks também estão disponíveis gratuitamente para os membros da rede. (UN WORLD FOOD PROGRAMME - Tradução própria)

Na figura 12 a troca de tokens em recursos humanitários sendo realizada:

Figura 12: Troca de tokens do projeto Building Blocks.

Fonte: Projeto Building Blocks

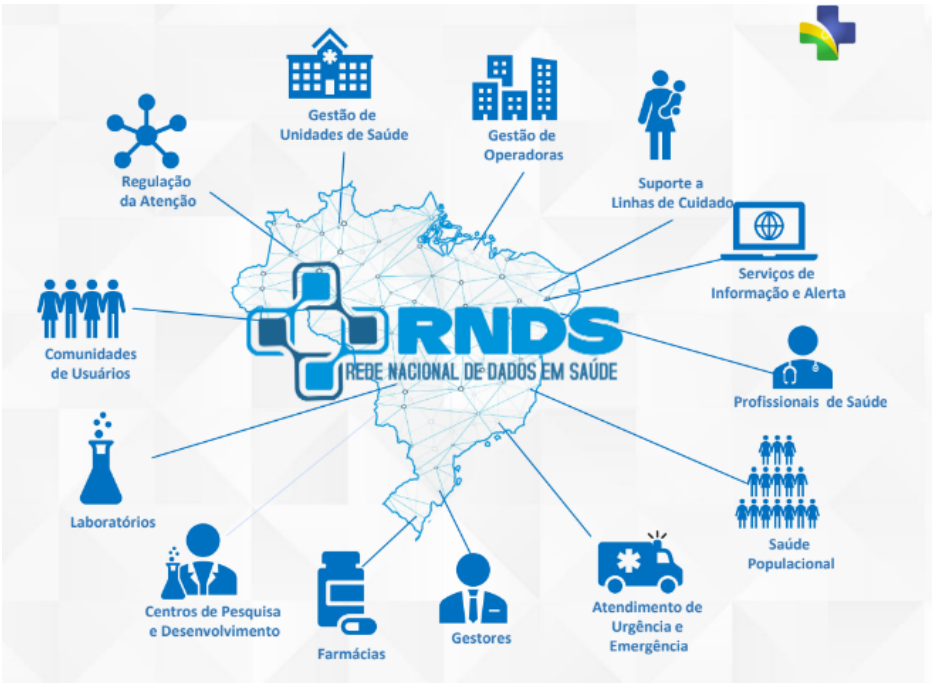
2.11.2 Saúde

O projeto brasileiro Conecte SUS (https://conectesus.saude.gov.br/home) unido com a empresa IAsis Health (https://health.iasis.tech/) munidos com a Estratégia de Saúde Digital para o Brasil (https://bvsms.saude.gov.br/), exploram o uso da Blockchain com o objetivo de ampliar o acesso à informação em saúde, fortalecendo o apoio à decisão clínica, vigilância em saúde, regulação, gestão e ensino e pesquisa.

A *RNDS (Rede Nacional de Dados em Saúde)* é a responsável por conectar os sistemas de informações das plataformas públicas e privadas espalhadas pelo território brasileiro.

Na figura 13 ilustração das conexões realizadas pelo projeto RNDS:

Figura 13: Conexão da Rede Nacional de Saúde



Fonte: Página do RNDS

Quando falamos de beneficiário, o usuário terá a oportunidade de continuar o seu atendimento seja na rede privada ou pública sem a burocracia de novos exames, pois a Blockchain junto com a base de dados irá guardar todas as informações do usuário, com a vantagem que o titular dos dados poderá permitir ou não que suas informações sejam visualizadas por determinados profissionais.

O projeto piloto utilizando Blockchain tem seu início no Estado de Alagoas, trazendo como principais benefícios o combate e controle de epidemias, melhoria no atendimento, acompanhamento de pacientes, eficiência na gestão do recurso público voltado à saúde e inovação na saúde.

Os benefícios da RNDS já podem ser acessados através do aplicativo Conecte SUS Cidadão para cidadãos comuns e Conecte SUS PRO para profissionais credenciados da rede nacional de saúde.

2.11.3 Combate ao Fake News

Com a ascensão da internet, das redes sociais e das mídias digitais, momentos importantes como as Eleições Eleitorais, podem ser influenciadas com a divulgação de notícias falsas. Vários escândalos vieram à tona nos últimos anos, como exemplo o caso das eleições de 2016, disputado pelos candidatos Hillary Clinton e Donald Trump, onde somente na rede social do Facebook 115 notícias falsas à favor do candidato Donald Trump foram compartilhadas por mais de 30 milhões de usuários (Aguiar apud SOLON; SIDDIQUI e ALLCOTT; GENTZKOW, 2017).

Diante destes fatos, sistemas como a utilizada pelo *Democracy Notary* (https://democracynotary.org/en.html), propoe validar documentos e relatórios oficiais publicados por entidades civis com base na Blockchain Emercoin (https://emercoin.com/en/). A ideia consiste em permitir que organizações e civis conhecidos e confiáveis façam o upload de conteúdos e os mesmos sejam registrados através de hashs imutáveis, garantindo segurança e segurança das informações.

Qualquer alteração em um conteúdo registrado resultará em um hash diferente, ou seja, a notícia será reconhecida como diferente da original.

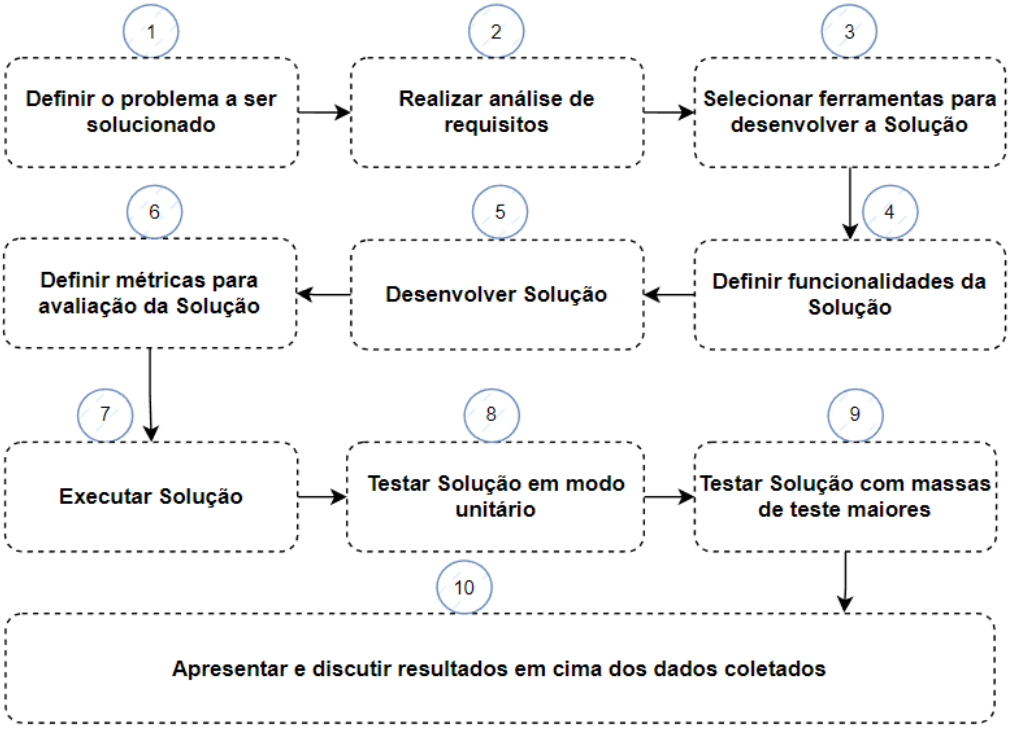
3. METODOLOGIA

Após a pesquisa bibliográfica realizada para entender os aspectos técnicos relacionados aos assuntos do tema, apresentaremos uma solução, onde mostraremos as etapas metodológicas, incluindo as ferramentas utilizadas para desenvolvimento do modelo de aplicação, a implementação e resultados coletados.

Na seção 3.1, apresentamos o problema a ser resolvido utilizando Blockchain. Na seção 3.2 analisaremos os requisitos para desenvolvimento da aplicação. Na seção 3.3, é mostrado os componentes necessários para implementação da solução. Na seção 3.4 definimos o tipo de padrão de desenvolvimento que utilizaremos para implementação da solução. Na seção 3.5 é apresentado uma visão geral da aplicação, junto com seu fluxo de funcionalidades. Na seção 3.6 demonstramos a implementação, execução e teste da aplicação. A seção de testes é subdividida em duas etapas: teste unitário e teste em massa. Por fim, na seção 4 apresentaremos uma análise dos resultados do experimento.

Na figura 14 o diagrama de fluxo dos procedimentos metodológicos:

Figura 14: Diagrama de fluxo dos procedimentos metodológicos



Fonte: Elaborado pelo autor (2022).

Devido às inúmeras possibilidades e complexidade que este trabalho pode levar, iremos nos limitar a apresentar um mecanismo que utilize alguns recursos da Blockchain, tais como contratos inteligentes, transações e blocos, diante disso iremos avaliar as transações após inclusão de informações na Blockchain.

3.1 Aplicação de Blockchain em um Sistema de Votação

O projeto a seguir tem como objetivo implementar e analisar um sistema de votação baseado em Blockchain, demonstrando através dos resultados gerados a segurança propiciada e desempenho, a fim de que as técnicas utilizadas neste estudo sirvam como base para futuros estudos de aplicações baseadas em Blockchain.

Conforme identificamos no referencial teórico, os problemas voltados à confiabilidade podem ser resolvidos através da Blockchain, pois por padrão os sistemas Blockchain garantem segurança e segurança das informações. Como proposta, iremos implementar um sistema de votação para diretoria em uma universidade (similar ao processo de votação da Universidade Estadual da Paraíba). Na seção a seguir serão apresentados os requisitos do sistema, em seguida será listado as ferramentas utilizadas para construção do mesmo.

3.2 Análise de Requisitos

Segundo Rezende (2005), a análise de requisitos tem por objetivo especificar as funcionalidades do sistema e suas restrições. Desta forma, iremos listar os requisitos necessários para implementar o sistema de votação utilizando Blockchain.

Sommerville (2003) define os requisitos como funcionais e não funcionais. Os requisitos funcionais são ações que o sistema deve oferecer, ou seja, são tarefas que o sistema deve realizar ao reagir a determinados estímulos. Já os requisitos não-funcionais, dizem respeito a restrições das funcionalidades do sistema. Abaixo listamos os requisitos:

3.2.1 Requisitos funcionais:

RF001 - Gerenciar eleitor: O sistema deve permitir o cadastro de eleitores;
RF002 - Gerenciar candidato: O sistema deve permitir o cadastro de candidatos;
RF003 - Gerenciar processo eleitoral - O sistema deve permitir o cadastro de um processo de eleição;
RF004 Gerar chaves - Deve ser possível criar uma chave privada para os usuários cadastrados;
RF005 - **Gerenciar acesso** - O usuário deve possuir um acesso único com login e senha para acessar o sistema de votação.
RF006 - **Gerenciar votação** - Deve possuir uma opção de aprovação dos eleitores recém cadastrados.

3.2.2 Requisitos não-funcionais:

RNF001 - O sistema deve ser acessado através de um navegador na web;
RNF002 - O sistema deve ser fácil de utilizar (usabilidade);
RNF003 - O sistema deve permitir auditoria dos blocos gerados pela Blockchain;
RNF004 - O sistema deverá se comunicar com o Banco de Dados MySQL para validação das chaves criadas.
RNF005 - O sistema deverá se comunicar com um Blockchain para registrar os votos dos eleitores.

Após análise dos casos de uso estudados, foram selecionadas as tecnologias para implementação do sistema de votação.

3.3 Ferramentas utilizados na implementação da solução

As tecnologias escolhidas para implementação da solução proposta, foram escolhidas pensando no custo zero de software. Todos os softwares se classificam com licença gratuita, acadêmica e “software de código livre”, aos quais impulsionam desenvolvedores a melhorarem suas bibliotecas e funcionalidades.

Abaixo agrupamos um diagrama com as tecnologias utilizadas para implementação da aplicação:

Figura 15: Ferramentas utilizadas para implementar a solução

Fonte: Elaborada pelo autor (2022)

3.3.1 Linguagens de programação

Como linguagem para Back-end (comunicação com o banco de dados e abstração das entidades do sistema) utilizamos o ***Java 8***, pois além de ser multiplataforma, a mesma permite a programação orientada a objetos, a qual podemos abstrair as entidades do nosso projeto, como: eleitor, candidato, voto. Além de contar com diversas bibliotecas que facilitam a comunicação com outros softwares. Também utilizamos o ***JDK*** ou Kit de desenvolvimento de Java, pois sua instalação é obrigatória para programar utilizando Java.

Como linguagem de Front-end (fornece uma interface amigável para o usuário) escolhemos o ***JavaScript***, por ser conhecida como a linguagem da web, facilitamos o acesso a interface gráfica e o Blockchain através das bibliotecas disponíveis nesta linguagem de programação. Utilizaremos o ***Node JS*** (https://nodejs.org/en/) para compilar a aplicação JavaScript. A comunicação estabelecida entre Back-end, Front-end e Docker foi via Api Rest, que utiliza um conjunto de operações HTTP: POST, GET, PUT e DELETE, sendo:

POST: utilizado com mais frequência para **criar** novos recursos.
GET: utilizado para **ler** (ou recuperar) uma representação de um recurso.
PUT: utilizado com mais frequência para recursos de **atualização**.
DELETE: utilizado para **excluir** um recurso identificado por um URI.

3.3.2 Framework

Adicionamos o ***Framework Spring*** para dar agilidade no processo de desenvolvimento da nossa aplicação Java Web. O Spring se encarrega de configurar servidor local, gerenciamento de dependências, configurações de bibliotecas e disponibiliza uma série de *annotations* (anotações) para utilizarmos em nossos serviços das classes Java.

3.3.3 Interface de Desenvolvimento

Utilizamos a plataforma ***Intellij IDEA*** (https://www.jetbrains.com/pt-br/idea/) como IDE (Interface de desenvolvimento), utilizando uma licença acadêmica. A IDE Eclipse (https://www.eclipse.org/downloads/) também pode ser utilizada para o desenvolvimento desta aplicação.

3.3.4 Armazenamento das informações

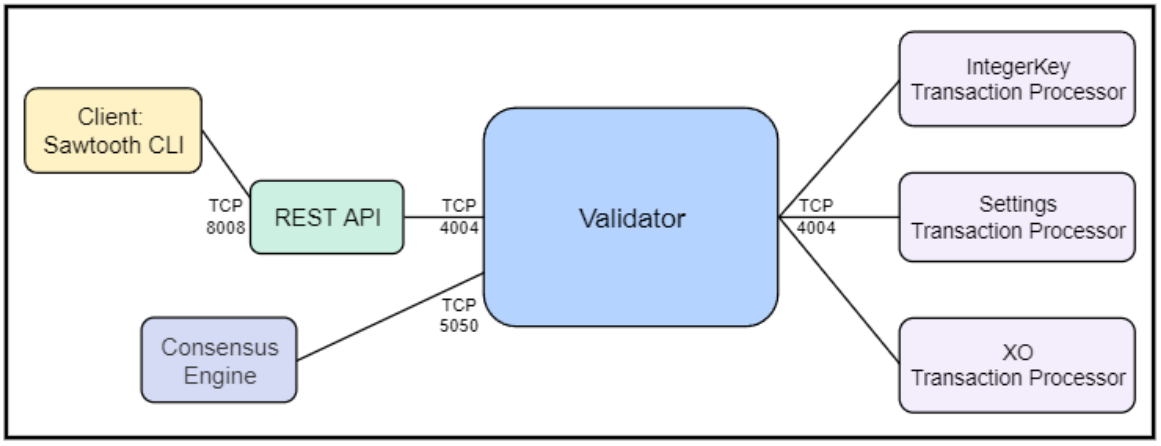
A título de estudo, iremos utilizar o Banco de Dados ***MySQL*** para acompanhar a criação das chaves públicas, usuários e votação. O diagrama UML (Linguagem de modelagem unificada) será apresentado na sessão de implementação do software.

3.3.5 Blockchain e Ledger

Como solução de livro-razão iremos utilizar a Blockchain ***Hyperledger Sawtooth*** (https://www.hyperledger.org/use/sawtooth) que é mantido pelo projeto Hyperledger Foundation e oferece mecanismos prontos para desenvolvedores e desta forma não se faz necessário desenvolver uma Blockchain do zero. A estrutura do projeto é focada em agilidade e basicamente fornece funcionalidades básicas como comunicação entre os nós da rede, armazenamento dos dados e arquitetura para conectar contratos inteligentes e algoritmos de consenso. O Hyperledger Sawtooth suporta uma variedade de algoritmos de consenso, incluindo tolerância a falhas bizantinas práticas (PBFT) e prova de tempo decorrido (PoET). Para esta aplicação utilizaremos o consenso **Devmode**, que é um algoritmo simplificado e

destina-se a testar aplicativos e é útil para processar transações e contratos inteligentes com rapidez, pois apenas um nó irá confirmar as requisições. Na figura 16 podemos visualizar uma ilustração da arquitetura Sawtooth de um único nó:

Figura 16: Arquitetura do Blockchain Sawtooth de um único nó.



Fonte: Hyperledger Sawtooth (2022)

Esta arquitetura de um único nó Sawtooth executa um Validador, uma API REST para conexão com aplicações externas, o mecanismo de consenso Devmode e três processadores de transação: IntegerKey (intkey-tp-python), Settings (sawtooth-settings) e XO (xo-tp-python), para processamento da lógica de negócios e comandos.

Mais informações sobre o Blockchain Sawtooth de um único nó pode ser acessada no link: https://sawtooth.hyperledger.org/docs/1.2/app_developers_guide/installing_sawtooth.html#docker

3.3.6 Virtualização da Blockchain

Iremos utilizar o ambiente do ***Docker*** *(https://www.docker.com/)* para virtualizar em contêiners a nossa aplicação de Blockchain. A documentação do Docker diz que *"Os contêineres são uma unidade padronizada de software que permite que os desenvolvedores isolem seu aplicativo de seu ambiente, resolvendo a dor de cabeça "funciona na minha máquina".* O ambiente Docker inclui um arquivo ***Docker Compose*** (veremos mais detalhes deste arquivo na fase de implementação)

que trata das etapas de configuração do ambiente, como gerar chaves e criar um bloco gênesis. O arquivo Compose também especifica as imagens de contêiner a serem baixadas do *Docker Hub* e as configurações de rede necessárias para que todos os contêineres se comuniquem corretamente. Mais informações do Docker e Docker Compose podem ser acessadas através do link: https://docs.docker.com/

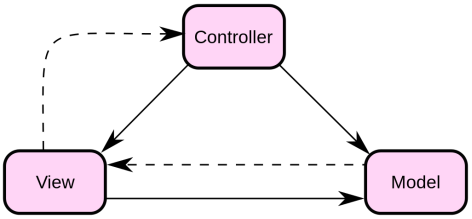
3.3.7 Versionamento do código

Para garantir a segurança do código e o versionamento da aplicação, utilizamos o GIT (https://git-scm.com/) que é um software de linha de comando voltado ao gerenciamento e versionamento do código e o GitHub (https://github.com/) que se trata de uma ferramenta online, ambas ferramentas gratuitas e amplamente utilizadas pela comunidade Dev. Projetos importantes estão armazenados no Github, tais como: JQuery, Node JS, Spring entre outros.

3.4 Padrão de Projeto MVC (Model-View-Controller)

Para simplificar o entendimento e desenvolvimento da aplicação, decidimos utilizar o modelo de Design Pattern ***MVC (Model-View-Controller)***, que é focado no reuso de código e separação de conceitos. Desta forma a aplicação se torna modular, sendo os conceitos divididos em, na figura 17 uma ilustração do padrão MVC:

Figura 17: Padrão MVC (Model-View-Controller)



Fonte: Google Imagens (2022)

Modelo: Define as regras de acesso e manipulação de dados. Funciona como uma ponte entre as camadas de Visão e Controle, consiste essencialmente na lógica da aplicação. (Gamma et al, 2020).

Visão: Define uma representação gráfica para a saída de dados, é onde os dados solicitados no modelo são exibidos. A na camada de visão que os usuários irão interagir com o sistema. (Gamma et al., 2020).

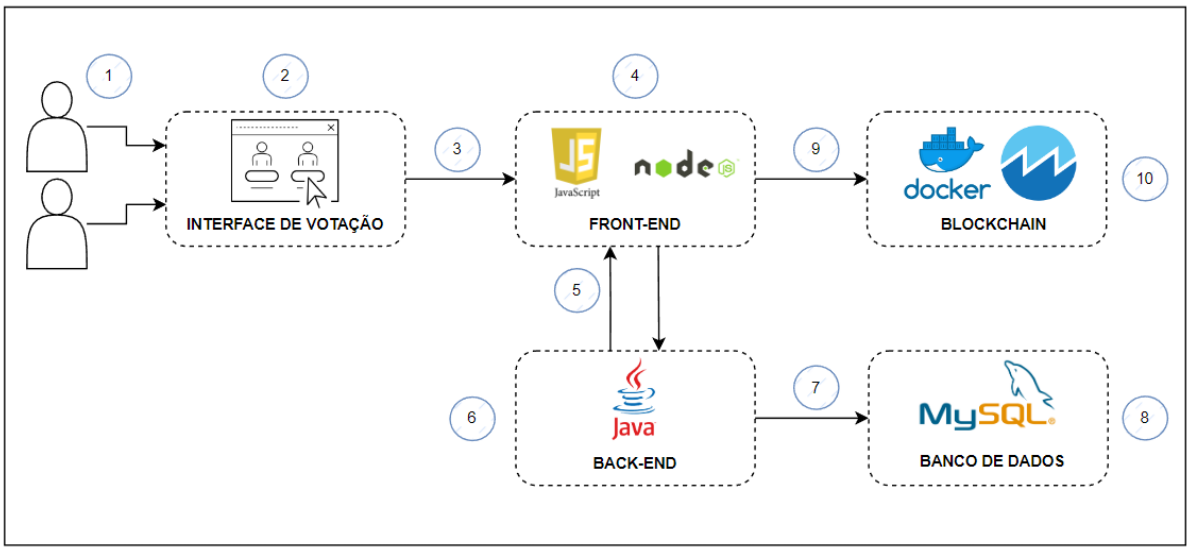
Controle: Faz a mediação da entrada e saída, comandando a visão e o modelo para serem alterados de forma apropriada conforme as solicitações do usuário na interface. O controle envia as informações para o modelo e para a visão. (Gamma et al., 2020).

3.5 Funcionalidades do Sistema

A proposta do estudo é implementar um sistema de votação que utilize a Blockchain como registrador de todos os votos realizados. Iremos utilizar o Blockchain Sawtooth, que nos entrega um sistema completo distribuído de Blockchain, independente da aplicação a ser utilizada. O Sawtooth fornece uma API de comunicação HTTP, a qual iremos utilizar do JavaScript para realizar tal comunicação, mediante entradas fornecidas pelo usuário no Front-end. O nosso sistema de Back-end irá controlar todos os fatores de autenticação, acesso e cadastro dos participantes e processos de eleição.

Na figura 18 é apresentado o fluxo que define o percurso do sistema de votação:

Figura 18: Fluxo de operação do sistema de votação



Fonte: Elaborada pelo autor (2022).

As setas no ponto (1) representam os eleitores, principais usuários do sistema de votação. Os usuários utilizam a interface de votação (2) fornecendo sua chave privada para escolher o candidato a ser votado. Após escolha e confirmação do candidato, o voto é transmitido via Api Rest (3) para o sistema de Front-End representado pelo servidor Node JS (4) ao qual transmite (5) e (9) os dados para o Back-end (6) e Blockchain (10). O Back-end mesmo antes de realizar a operação de salvar os dados no Banco de Dados (7), valida todo o acesso dos usuários, assim como apresenta ao Front-end todas as entidades necessárias para a interface do sistema de votação. A comunicação entre a aplicação Javascript (4) e o Blockchain (10) é realizada via Api Rest (9), e neste passo o voto é registrado na Blockchain (10) de forma imutável e transparente a rede.

3.6 Desenvolvimento da Solução, Métricas e Execução

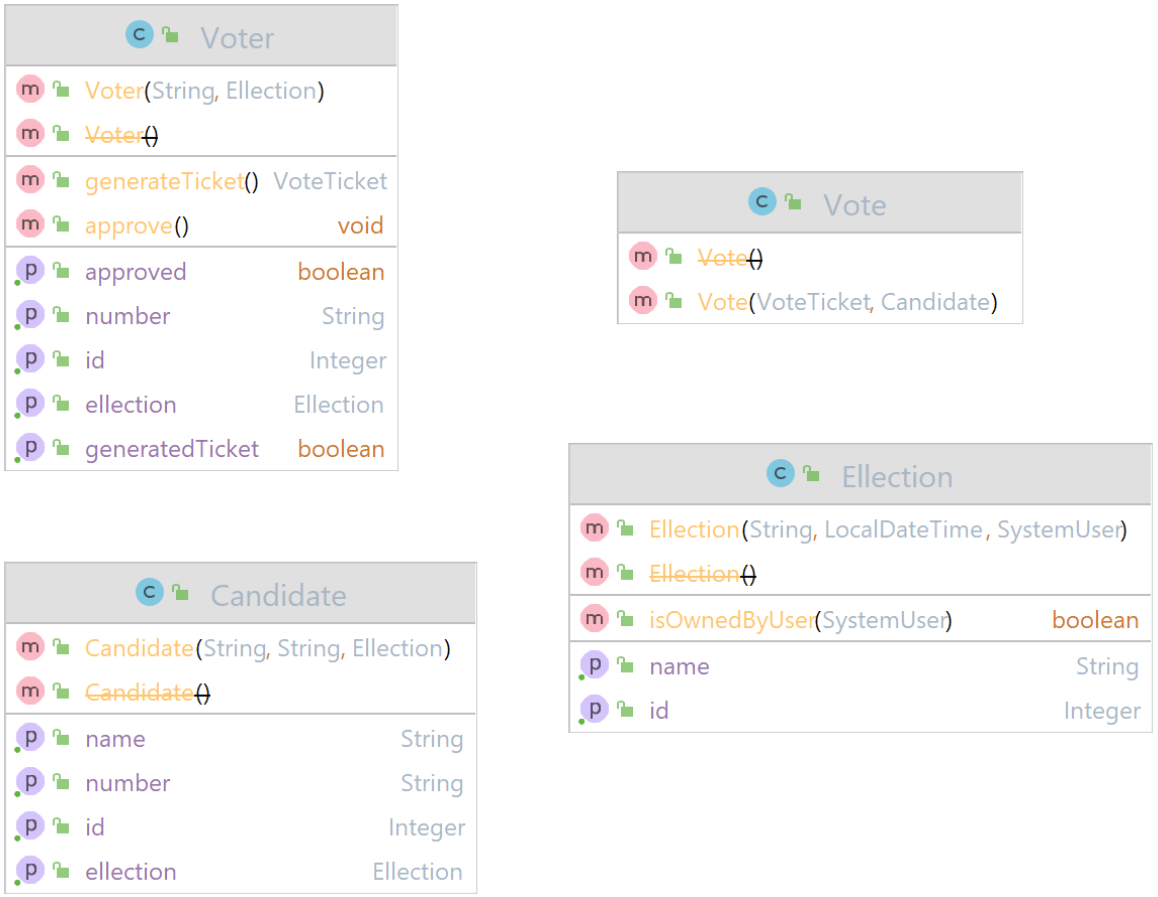
Nesta seção iremos descrever a implementação da solução de votação (back-end e front-end) e a configuração da Blockchain Sawtooth.

3.6.1 Implementação do Back-end

Conforme requisições apresentadas, evoluímos na codificação apresentando as principais classes *.java* responsáveis pelas entidades da nossa aplicação, a implementação das principais classes pode ser visualizada no link do Apêndice A.

Abaixo os principais **modelos (Models)** da aplicação gerados pelo Diagrama do Intellij, com suas propriedades e métodos, sendo *Voter* o eleitor que irá votar no sistema de eleição, *Candidate* o candidato da eleição, *Vote* o voto do eleitor e *Ellection* a eleição em si. Na figura 19 visualizamos os modelos do projeto:

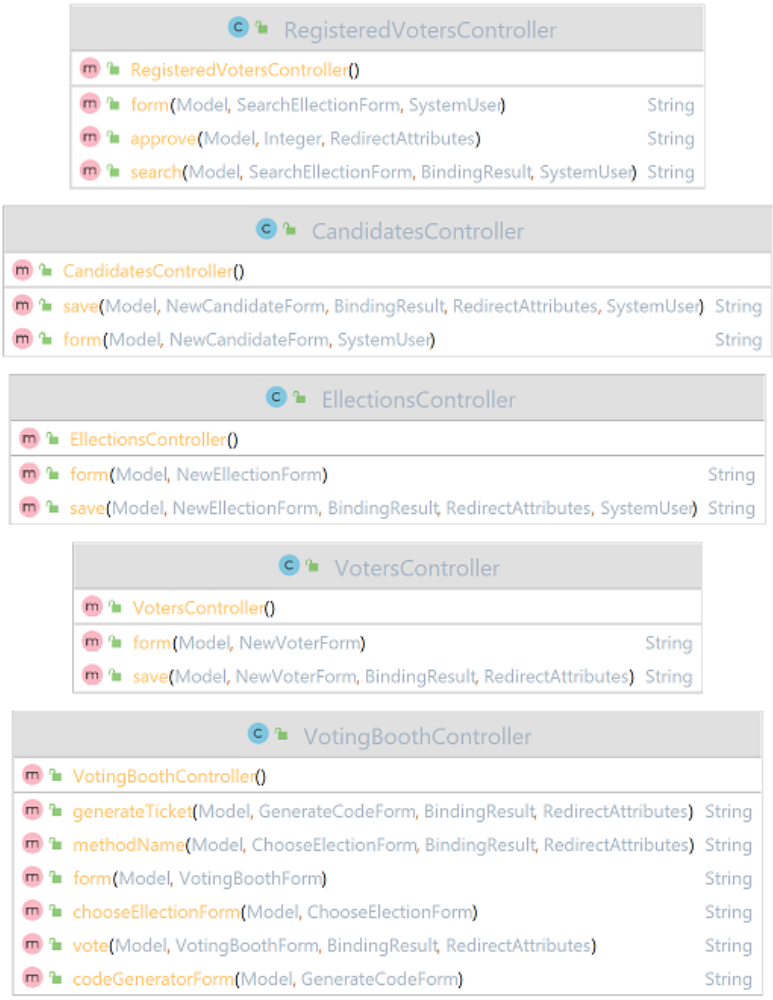
Figura 19: Classes Modelos da aplicação



Fonte: Elaborado pelo autor (2022).

Abaixo as classes de **controle (Controllers)** responsáveis por receber as requisições da interface, sendo *RegisteredVotersController* é responsável por controlar o registro dos votos realizados na interface da solução, além de resolver a solicitação de aprovação para um novo eleitor. A classe *CandidatesController* é responsável por controlar as requisições referentes à entidade candidato. A classe *EllectionsController* controla as requisições de cadastro para novas eleições. *VotersController* controla o cadastro de novos eleitores. Já a classe *VotingBoothController* é responsável por gerenciar os tickets (chaves privadas) dos eleitores, sendo responsável também por testar no momento da votação se o eleitor está liberado ou não para realizar a votação. Na figura 20 podemos visualizar as principais Controllers da aplicação:

Figura 20: Classes Controllers da aplicação

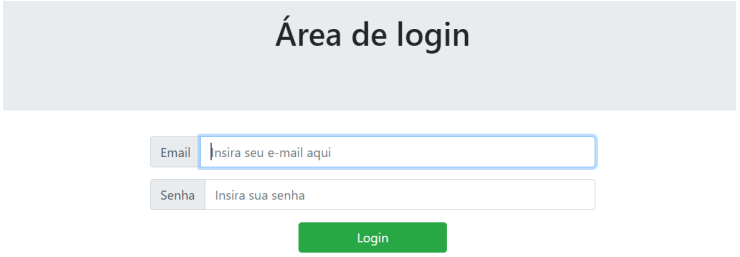


Fonte: Elaborada pelo autor (2022)

Finalizando o modelo MVC (Model-View-Controller), criamos interfaces responsáveis pela View de cada entidade: Candidate, Voter e Election.

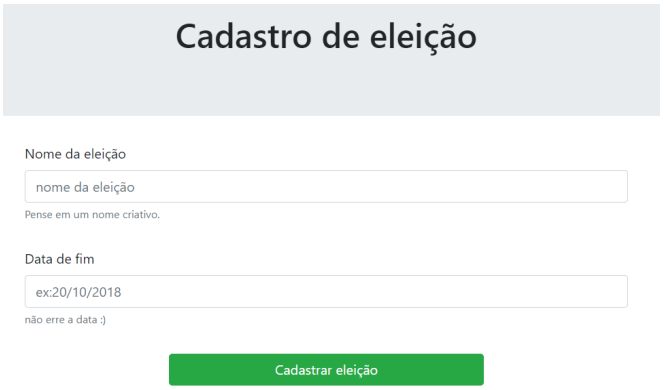
As interfaces compiladas nas imagens de 21 à 27:

Figura 21: Tela de login da aplicação (View)



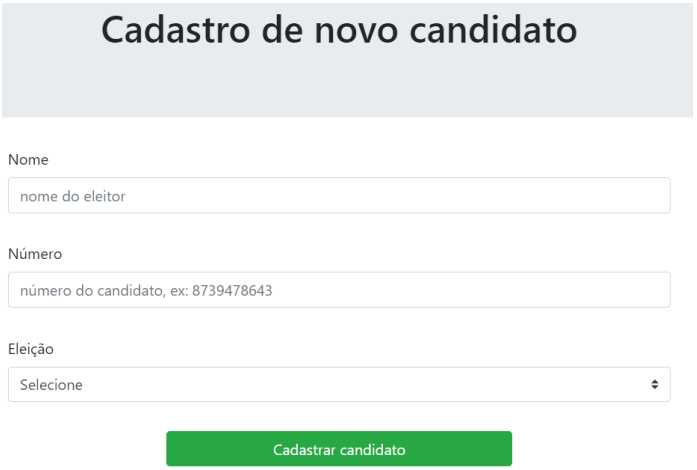
Fonte: Elaborado pelo autor (2022).

Figura 22: Tela de cadastro da eleição (View)



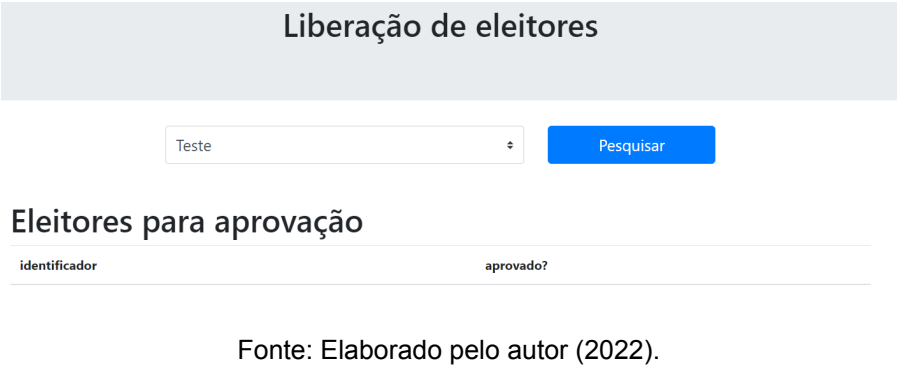
Fonte: Elaborado pelo autor (2022).

Figura 23: Tela de cadastro do candidato (View)



Fonte: Elaborado pelo autor (2022).

Figura 24: Tela de aprovação dos eleitores (View)

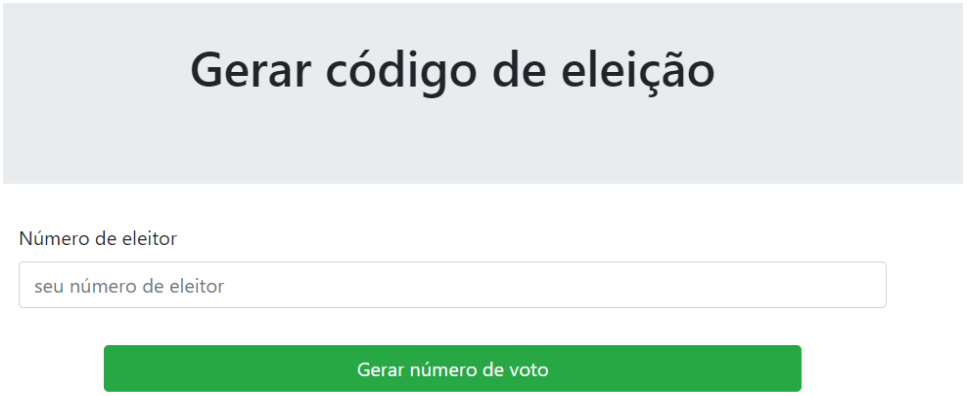


Fonte: Elaborado pelo autor (2022).

Figura 25: Tela de cadastro do eleitor (View)

Fonte: Elaborado pelo autor (2022).

Figura 26: Tela para gerar token privado para o eleitor (View)



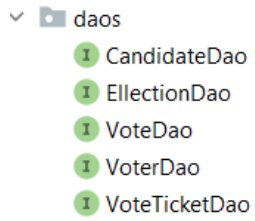
Fonte: Elaborado pelo autor (2022).

Figura 27: Tela de votação para o eleitor (View)

Fonte: Elaborado pelo autor (2022).

Além das implementações MVC, adicionamos Interfaces DAO (Data Access Object), que segundo DevMedia (2014) são interfaces que auxiliam a troca de informações com o Banco de Dados. Na figura 28 pacote de interfaces DAO implementadas:

Figura 28: Pacote de Interfaces DAO da aplicação



Fonte: Elaborado pelo autor (2022).

3.6.2 Implementação do Front-end

A nossa aplicação JavaScript que executa sobre o Node JS será responsável por fazer o acesso necessário para registrar o voto na Blockchain Sawtooth, e para isto utilizaremos o framework Restify (http://restify.com/) para criar uma Api que possamos acessar via HTTP.

- O arquivo **index.js** é responsável por receber os votos e registrar na Blockchain;
- O arquivo **client.js** irá fornecer um método - registerBlockchain - para registro do voto (payload) e a função sendToSawtoothApi() envia um array de bytes como registro para a Blockchain;
- O arquivo ***processor.js*** funcionará como tratamento do nosso contrato inteligente (smart contract), utilizando da função *TransactionProcessor()* para validação do nosso handler (contrato inteligente);
- O arquivo ***voteHandler.js*** codifica a nossa função de contrato inteligente a qual podemos criar diversas validações e restrições para execução do contrato inteligente;
- O arquivo infra.js implementa todas os métodos fornecidos pelo SDK Sawtooth (https://openbase.com/js/sawtooth-sdk/documentation), o

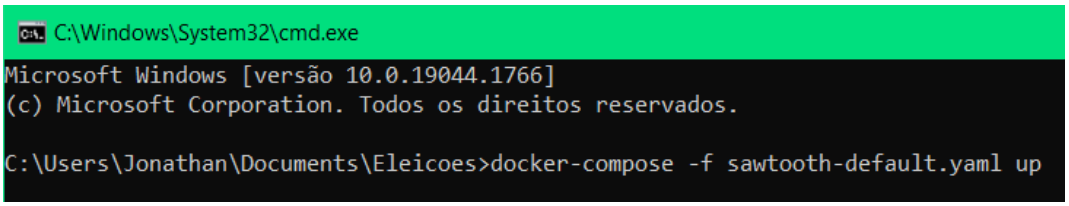
código é disponibilizado pela documentação e basta seguirmos o fluxo informado;

A implementação dos arquivos citados acima podem ser visualizados no link do Apêndice A e algumas implementações como o contrato inteligente no Apêndice B.

3.6.3 Implementação da Blockchain Sawtooth

Para implementar o serviço de Blockchain iremos utilizar o arquivo de configuração Docker Compose disponibilizado pelo projeto Hyperledger (https://github.com/hyperledger/sawtooth-core/blob/1-2/docker/compose/sawtooth-default.yaml). Para executar o arquivo sawtooth-default.yaml devemos baixar e instalar o Docker (https://docs.docker.com/desktop/install/windows-install/), depois de executar o Docker abrimos o CMD (Promp de Comando), acessamos a pasta do arquivo .yaml baixado e executamos o comando abaixo na figura 29:

Figura 29: Execução do comando para instanciar o serviço do Sawtooth

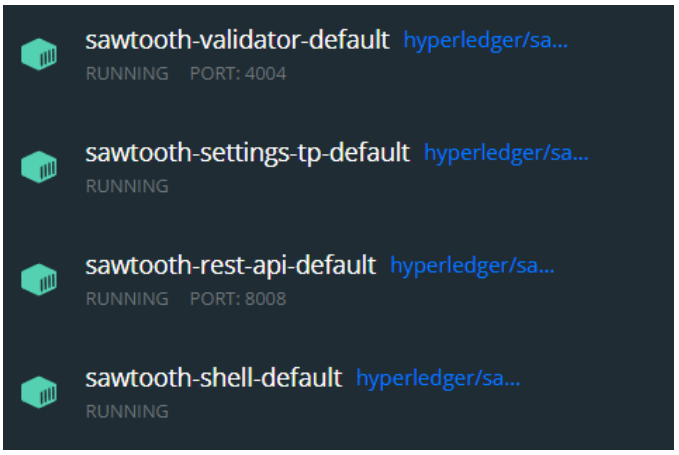


Fonte: Elaborado pelo autor (2022).

Para utilizar o algoritmo de consenso PoET basta executar o comando *sawtooth-default-poet.yaml,* para utilizar o consenso PBFT utilize o comando *sawtooth-default-pbft.yaml.*

Após execução do comando, no docker poderemos identificar alguns contêineres criados:

Figura 30: Contêineres criados na execução do Docker Compose



Fonte: Elaborado pelo autor (2022).

3.6.4 Definição de métricas para Avaliação

Para esta etapa realizaremos uma avaliação empírica do tipo quantitativa, onde partiremos da hipótese de que vários eleitores registraram votos, para isto precisaremos configurar nos testes quantidades adequadas para o envio dos votos.

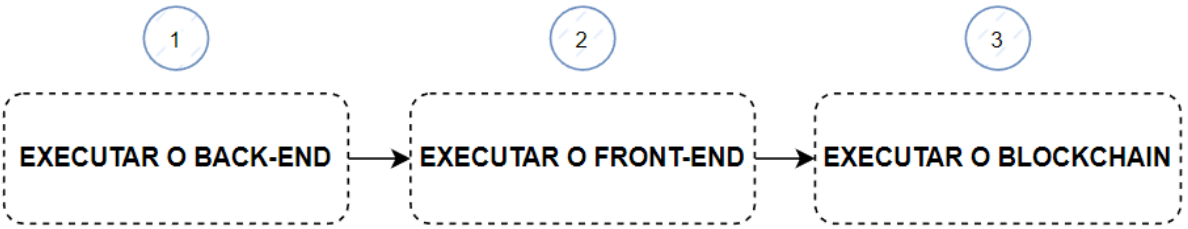
A solução foi avaliada com o objetivo de verificar o estado de segurança das informações enviadas e o desempenho quando submetido a diferentes cargas de trabalho. Para a avaliação da segurança faremos testes unitários enviando votos com o mesmo usuário, demonstrando que os blocos uma vez registrados não podem ser alterados. Para o teste de desempenho, faremos vários testes de massa, ao qual foram escolhidas variáveis relacionadas aos votos, sendo as variáveis: *quantidade de votos*, *tempo de execução (latência)* e *tipo de requisição* HTTP. A quantidade de requisições do tipo POST simulam o envio de votos para a Blockchain. Faz-se necessário um teste de carga pois leva-se em consideração que quanto mais requisições, menor o desempenho da blockchain.

Utilizaremos métodos HTTP do tipo POST para testar o sistema em teste de massa. O método POST irá salvar os votos na Blockchain. Utilizaremos o software JMeter para simular o envio de grandes quantidades de votos. Informações sobre o JMeter podem ser encontradas no link https://jmeter.apache.org/

3.6.5 Execução da aplicação

Para acessar a interface da aplicação devemos executar os processos de acordo com o fluxo da figura 31:

Figura 31: Fluxo de execução da aplicação



Fonte: Elaborado pelo autor (2022).

3.6.5.1 Executar o Back-end

Primeiramente precisamos inicializar a estrutura de servidor da aplicação, ou seja, o back-end. Para inicializar o back-end basta clicar no “play” (figura 32) com o projeto carregado na IDE na classe *“Boot”*, pois o Spring Boot da aplicação irá tratar de toda a infraestrutura necessária para acessarmos o projeto via browser, além de criar a base dados MySQL:

Figura 32: Classe de inicialização da aplicação

```
@SpringBootApplication
public class Boot extends SpringBootServletInitializer{

    public static void main(String[] args) { SpringApplication.run(Boot.class, args); }
}
```

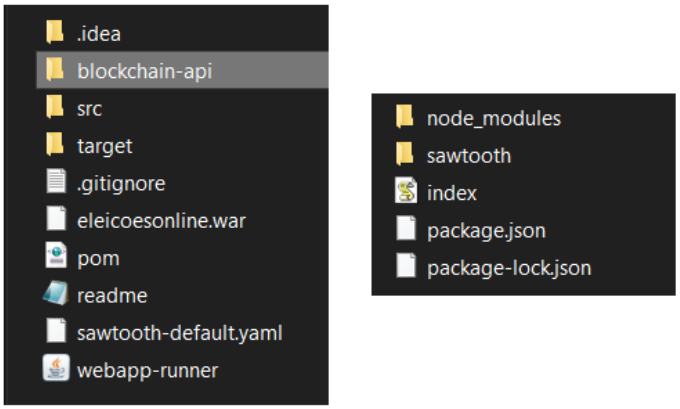
Fonte: Elaborado pelo autor (2022).

Para acessar o sistema utilizaremos o link http://localhost:8088/login

3.6.5.2 Executar o Front-end

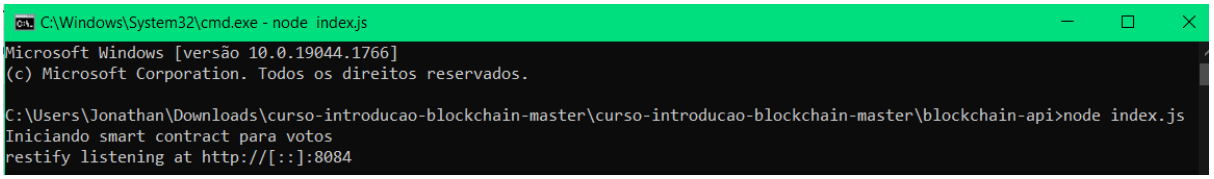
Para inicializar a estrutura front-end precisamos entrar na pasta do projeto (figura 33), abrir a pasta “Blockchain-api” que é onde está localizado a aplicação JavaScript, abrir o prompt de comando e executar o comando *node index.js (figura 34)*:

Figura 33: Estrutura de pastas do projeto front-end



Fonte: Elaborado pelo autor (2022).

Figura 34: Inicializando aplicação front-end da aplicação



Fonte: Elaborado pelo autor (2022).

Após execução poderemos acessar a Api do Front-end através do link http://localhost:8084/

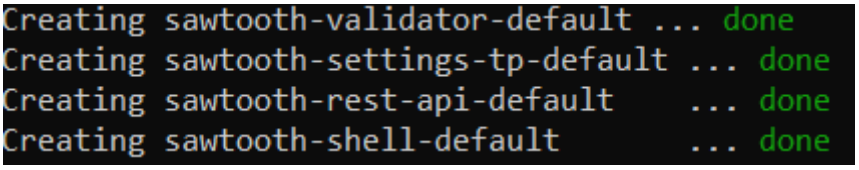
3.6.5.3 Executar o Blockchain

Conforme vimos na seção 3.6.3 basta acessar a pasta do arquivo .yaml e executar o comando sawtooth-default.yaml no CMD.

O comando executado acima define alguns componentes, dentre elas: um único validador usando o consenso do Devmode, uma API REST conectada ao validador, o processador de transações Configurações (sawtooth-settings), o

processador de transações IntegerKey (intkey-tp-python), o processador de transações XO (xo-tp-python) e um contêiner de cliente (shell) para executar comandos do Sawtooth conforme figura 35:

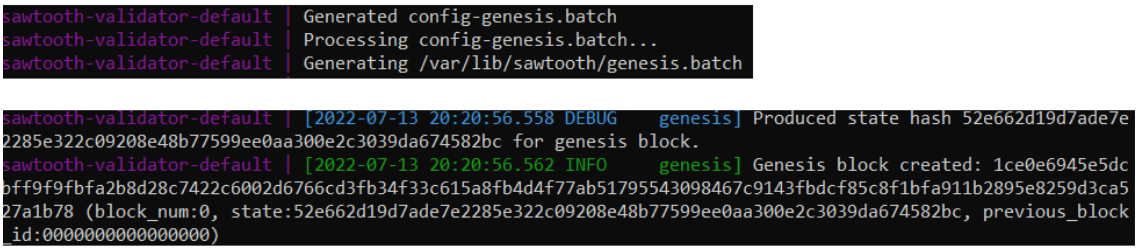
Figura 35: Criação de estrutura de execução do Sawtooth



Fonte: Elaborado pelo autor (2022).

A execução do *sawtooth-default.yaml* também é responsável por criar o arquivo config-genesis.batch (figura 36) que irá gerar o bloco gênesis da nossa aplicação:

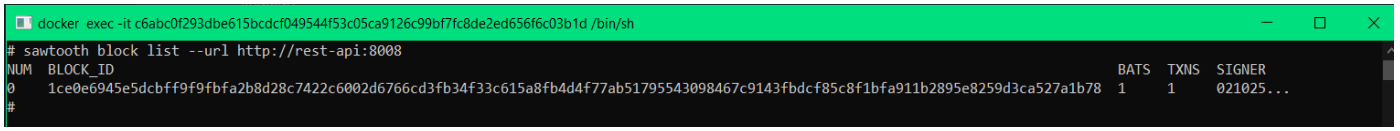
Figura 36: Criação do bloco gênesis da aplicação



Fonte: Elaborado pelo autor (2022).

Na figura 37 podemos ver o comando ***sawtooth block list*** que lista os blocos criados e seu estado:

Figura 37: Bloco criado na execução inicial do Sawtooth



Fonte: Elaborado pelo autor (2022).

Para exibir os detalhes do bloco utilizamos o comando *sawtooth block show --url http://rest-api:8008 {BLOCK_ID},* onde BLOCK_ID é o ID do bloco que desejamos visualizar os detalhes na figura 38:

Figura 38: Detalhes de um bloco Sawtooth


```
batches:
- header:
    signer_public_key: 0276023d4f7323103db8d8683a4b7bc1eae1f66fbbf79c20a51185f589e2d304ce
    transaction_ids:
    - 24b168aaf5ea4a76a6c316924a1c26df0878908682ea5740dd70814e7c400d56354dee788191be8e2839
  header_signature: a93731646a8fd2bce03b3a17bc2cb3192d8597da93ce735950dccbf0e3cf0b005468fa
  transactions:
  - header:
      batcher_public_key: 0276023d4f7323103db8d8683a4b7bc1eae1f66fbbf79c20a51185f589e2d304
      dependencies: []
      family_name: sawtooth_settings
      family_version: '1.0'
...
header:
  batch_ids:
  - a93731646a8fd2bce03b3a17bc2cb3192d8597da93ce735950dccbf0e3cf0b005468fadb94732e013be0bc
  block_num: 3
  consensus: RGV2bW9kZQ==
  previous_block_id: 042f08e1ff49bbf16914a53dc9056fb6e522ca0e2cff872547eac9555c1de2a6200e6
  signer_public_key: 033fbed13b51eafaca8d1a27abc0d4daf14aab8c0cbc1bb4735c01ff80d6581c52
  state_root_hash: 5d5ea37cbbf8fe793b6ea4c1ba6738f5eee8fc4c73cdca797736f5afeb41fbef
header_signature: ff4f6705bf57e2a1498dc1b649cc9b6a4da2cc8367f1b70c02bc6e7f648a28b53b5f6ad7
```

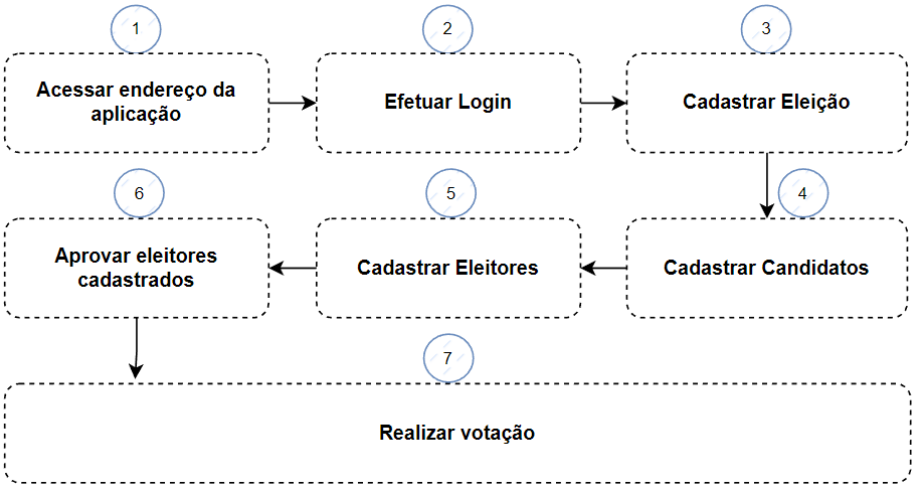

Fonte: Elaborado pelo autor (2022).

Pronto, agora temos todas as partes executando. Vamos aos testes!

3.6.6 Testando a solução em modo unitário

Para testar a solução em modo unitário devemos seguir o fluxo de interação da figura 39:

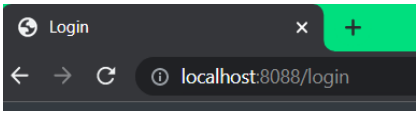
Figura 39: Fluxo de interação da solução



Fonte: Elaborado pelo autor (2022).

Após executar os passos 3.6.4.1, 3.6.4.2 e 3.6.4.3, para acessar a interface da aplicação (1) devemos acessar o endereço http://localhost:8088/login no navegador do nosso computador conforme figura 40:

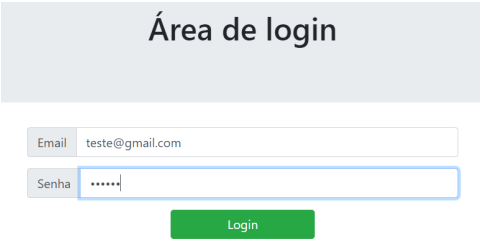
Figura 40: Endereço local da aplicação



Fonte: Elaborado pelo autor (2022).

Após acessar o endereço da aplicação, devemos digitar o login e senha (2) de autenticação do sistema conforme figura 41 (a título de testes, inserimos diretamente no código o e-mail: teste@gmail.com e a senha: 123456).

Figura 41: Tela de login da aplicação web



Fonte: Elaborado pelo autor (2022).

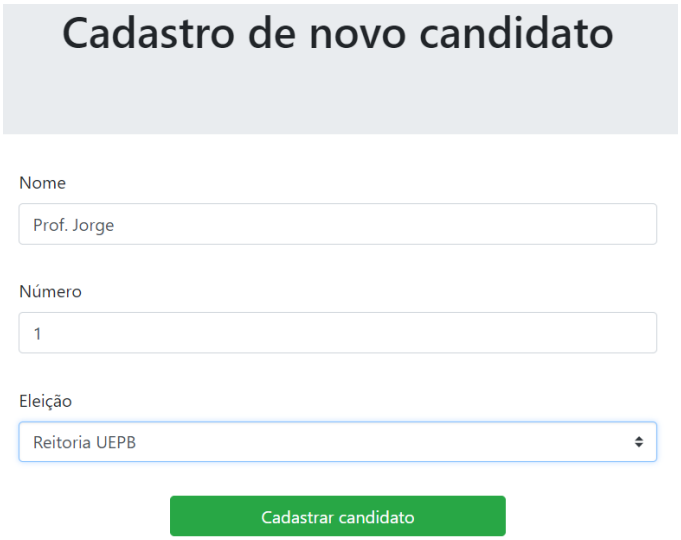
Na sequência (figura 42) podemos cadastrar uma nova Eleição (3), fornecendo nome, data e hora de fim:

Figura 42: Cadastro de uma nova eleição

Fonte: Elaborado pelo autor (2022).

O próximo passo será cadastrar os Candidatos a Eleição (4), informando o nome do candidato, número que representa o candidato e Eleição que ele irá participar como candidato, veja a figura 43:

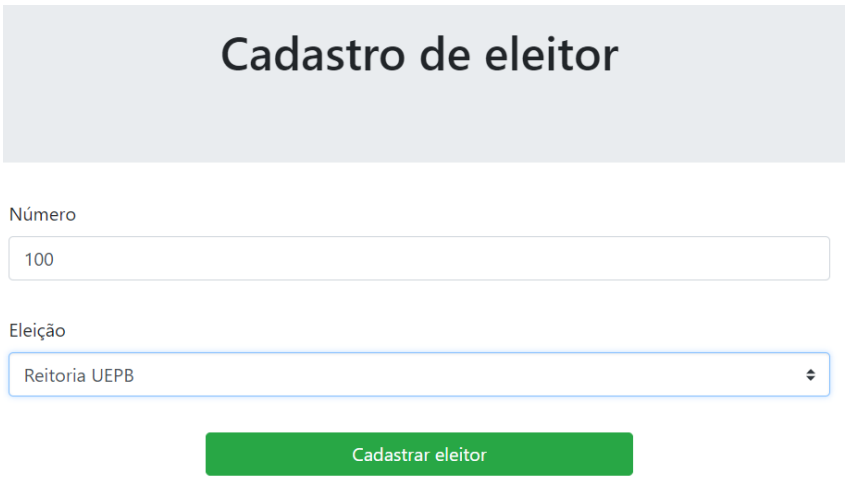
Figura 43: Cadastrando um novo candidato



Fonte: Elaborado pelo autor (2022).

Agora precisamos cadastrar o eleitor (figura 447), informando seu código e a eleição que o mesmo poderá participar:

Figura 44: Cadastrando um novo eleitor

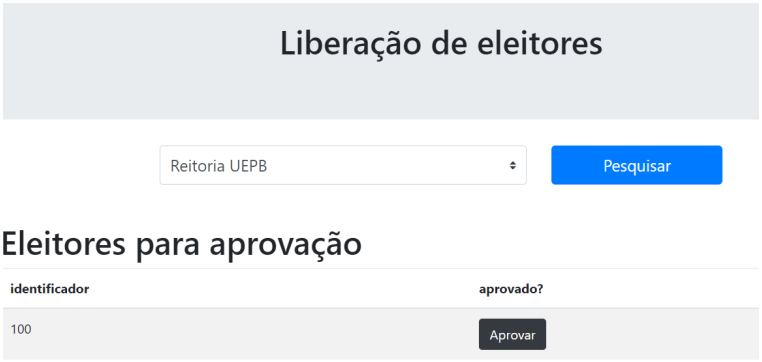


Fonte: Elaborado pelo autor (2022).

Por motivos de simulação de privacidade, não iremos solicitar o nome do eleitor.

Neste momento o eleitor ainda não pode votar, precisamos aprovar e gerar uma chave privada para o mesmo, conforme figura 45. Segue tela de aprovação simulando o gerente do sistema:

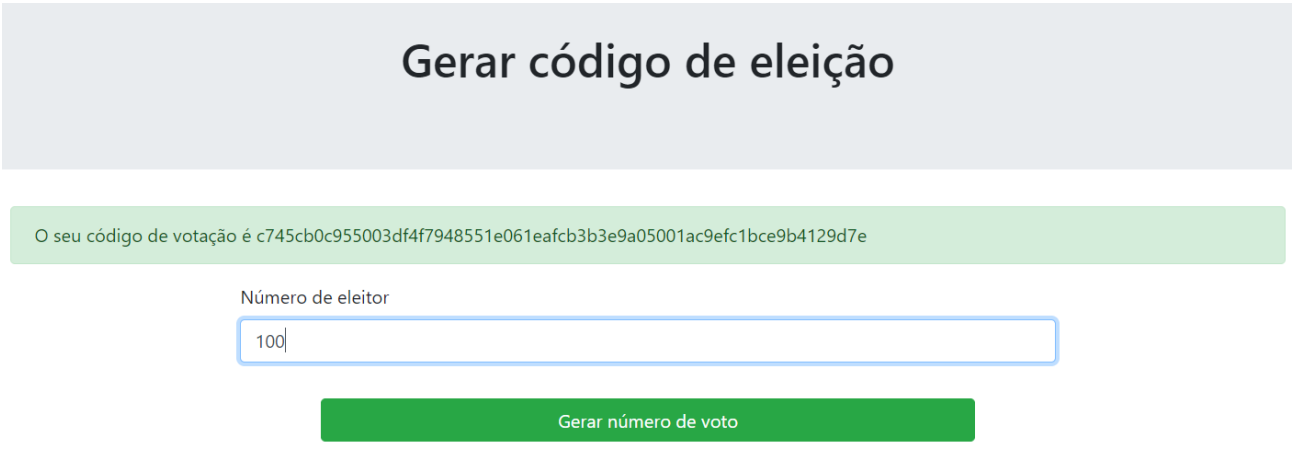
Figura 45: Aprovando um novo eleitor



Fonte: Elaborado pelo autor (2022).

Após realizar a aprovação do eleitor, é necessário que o eleitor gere seu token de votação, informando apenas seu código de eleitor (figura 46). O token de votação simula uma chave privada, sendo gerado um hash único ao qual somente o titular deve ter acesso. Como medida didática, salvaremos esse token no banco de dados para acesso posterior:

Figura 46: Tela de geração do token privado para o eleitor

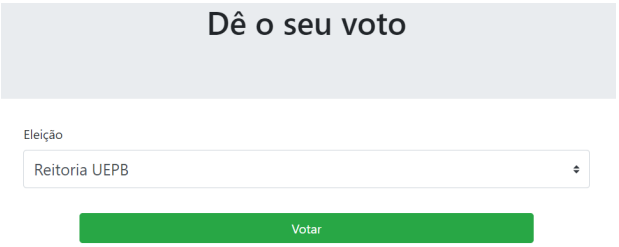


Fonte: Elaborado pelo autor (2022).

Com o código privado em mãos poderemos registrar nosso voto. Até o momento o Java tratou todos os cadastros realizados, a partir deste passo o Javascript se encarregará de verificar o contrato inteligente e registrar o voto na Blockchain Sawtooth. Para registrar o voto acessamos a url: http://localhost:8088/votingbooth/choose/election/form

Selecionamos a eleição e clicamos em votar conforme figura 47:

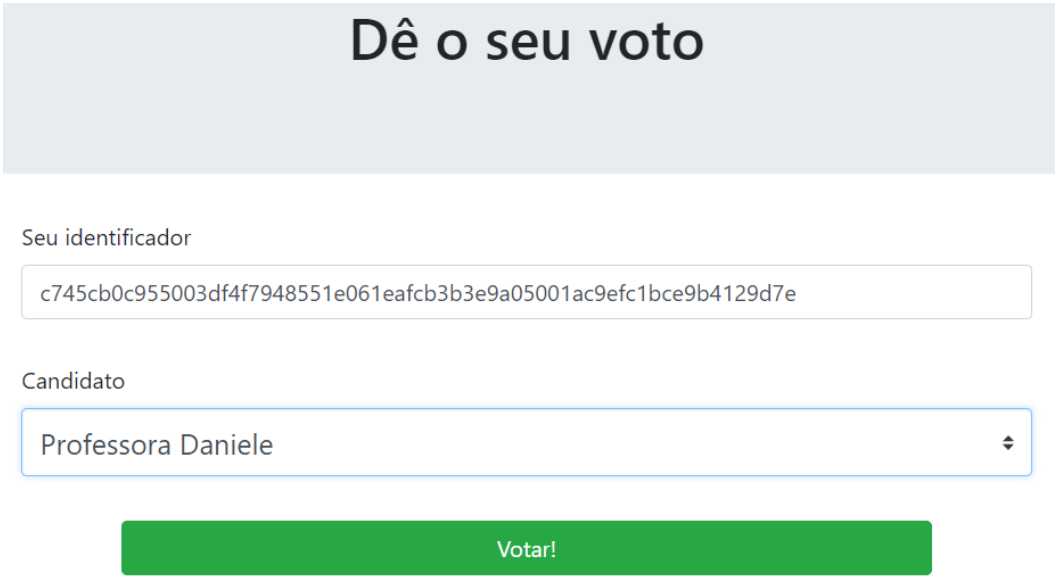
Figura 47: Selecionando uma eleição



Fonte: Elaborado pelo autor (2022).

No próximo passo (figura 48) informamos a chave privada e selecionamos o candidato ao qual desejamos registrar nosso voto:

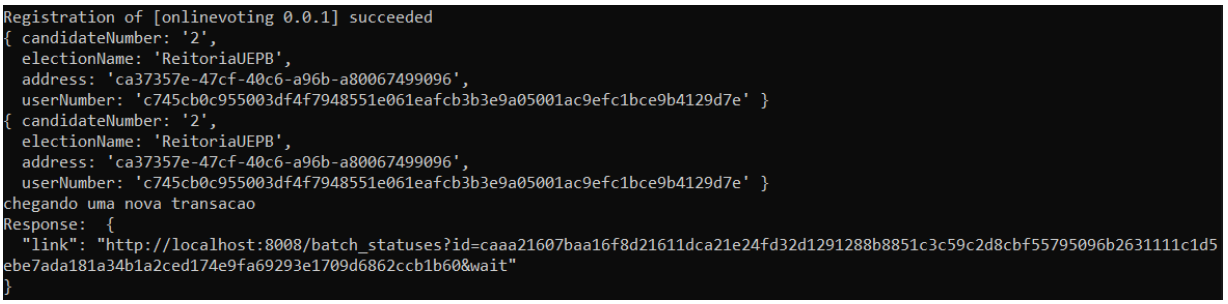
Figura 48: Informando chave privada e escolhendo candidato



Fonte: Elaborado pelo autor (2022).

Ao clicar em “Votar!” estaremos enviando uma request via Api Rest para o Blockchain Sawtooth e nosso voto será registrado na Blockchain:

Figura 49: Requisição enviada através da Api Rest Javascript

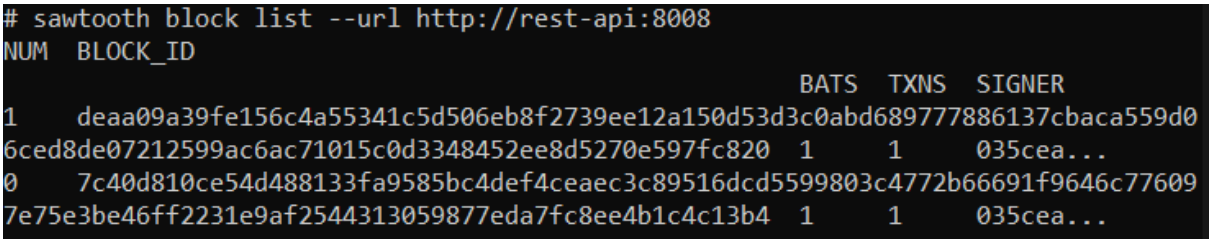


Fonte: Elaborado pelo autor (2022).

Neste momento uma nova requisição é recebida na Blockchain, um novo bloco é verificado e criado utilizando o consenso devmode.

Ao acessar o shell no contêiner *sawtooth-shell-default* e utilizar o comando *sawtooth block list --url http://rest-api:8008* poderemos identificar dois blocos criados, sendo **NUM 0** o bloco gênesis e **NUM 1** o bloco criado para o novo voto registrado, conforme figura 50:

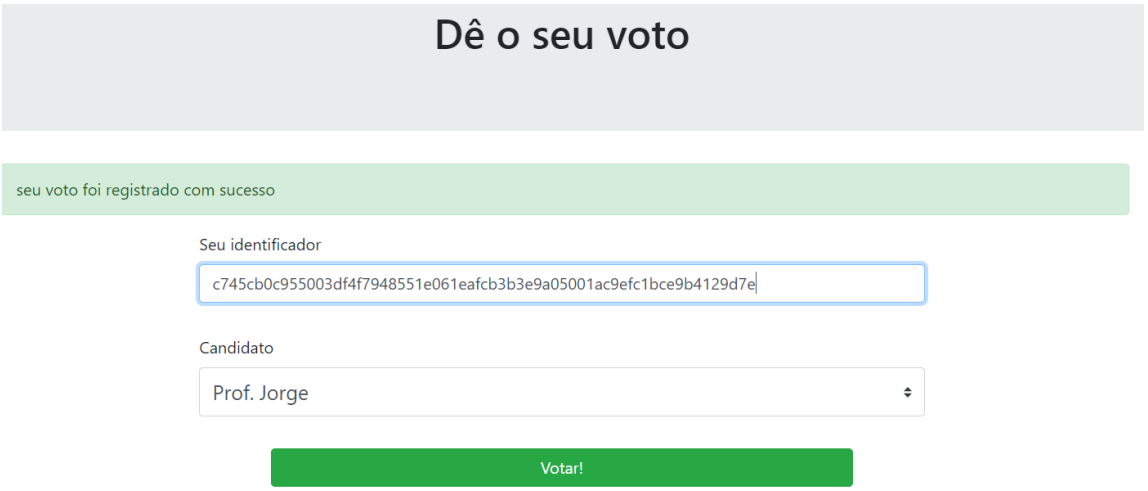
Figura 50: Blocos criados no Blockchain sawtooth



Fonte: Elaborado pelo autor (2022).

Como prova de segurança, iremos registrar um novo voto com o mesmo usuário, porém selecionando um candidato diferente. Na vida real isto não seria possível, pois cada eleitor somente poderia votar uma única vez, no entanto, para que possamos provar a segurança do voto registrado, iremos liberar esta funcionalidade. Na figura 66 cadastramos um novo voto, desta vez para o Prof. Jorge:

Figura 51: Registro de um novo voto com a mesma chave privada



Fonte: Elaborado pelo autor (2022).

Após registrar o voto vamos visualizar novamente a saída do comando *sawtooth block list --url http://rest-api:8008* no shell do Sawtooth (figura 52):

Figura 52: Novo bloco criado no Blockchain sawtooth

```
# sawtooth block list --url http://rest-api:8008
NUM  BLOCK_ID                                                        BATS  TXNS  SIGNER
2    71d6ce7de24c150bef9c8cb8ed0c27657a6ce86229e66c2c872bc2695181b7ba777f317bc47f
fbc422476013179fb21430178b04ce94151ff91e94af89ebc8be  1     1     035cea...
1    deaa09a39fe156c4a55341c5d506eb8f2739ee12a150d53d3c0abd689777886137cbaca559d0
6ced8de07212599ac6ac71015c0d3348452ee8d5270e597fc820  1     1     035cea...
0    7c40d810ce54d488133fa9585bc4def4ceaec3c89516dcd5599803c4772b66691f9646c77609
7e75e3be46ff2231e9af2544313059877eda7fc8ee4b1c4c13b4  1     1     035cea...
#
```

Fonte: Elaborado pelo autor (2022).

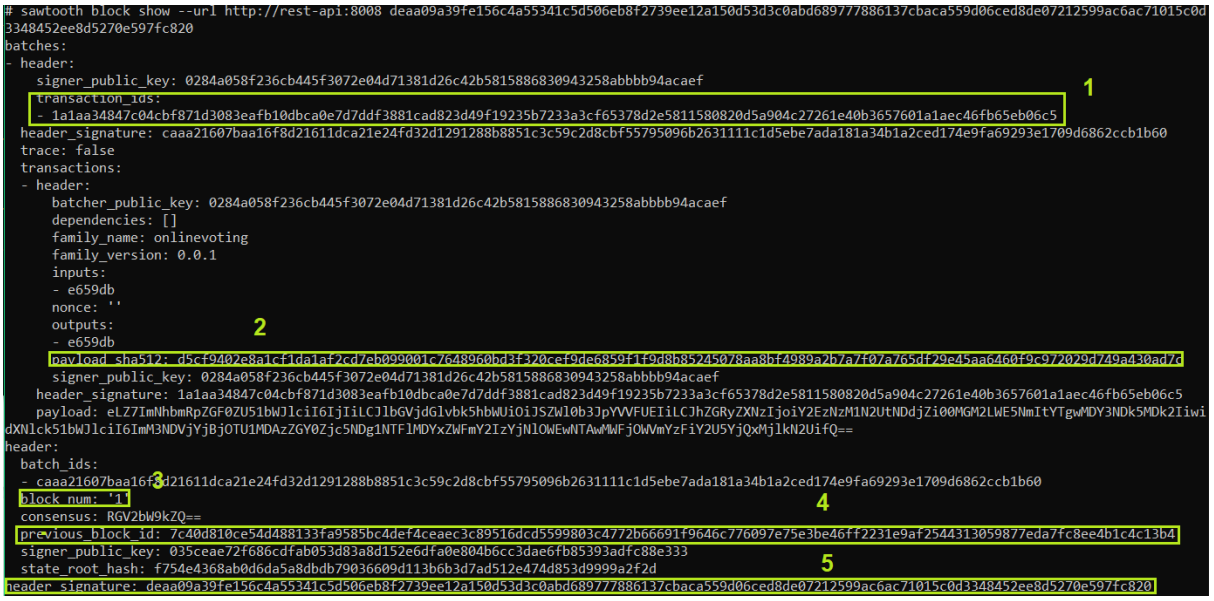
Como podemos perceber, um novo bloco (NUM 2) foi criado para registrar o novo voto do mesmo eleitor, isto prova que uma vez registrado um dado na Blockchain, o mesmo não pode ser alterado. Outra característica importante é que de forma transparente podemos verificar todos os votos registrados, evitando fraudes e polêmicas no sistema de votação.

Indo um pouco mais afundo, vamos acessar o bloco 1 (figura 53) que corresponde ao registro do primeiro voto através do comando *sawtooth block show --url http://rest-api:8008 {block}*

Onde {block} é o endereço do nosso bloco criado assim que solicitamos a requisição:

deaa09a39fe156c4a55341c5d506eb8f2739ee12a150d53d3c0abd689777886137cba
ca559d06ced8de07212599ac6ac71015c0d3348452ee8d5270e597fc820

Figura 53: Detalhes do bloco 1



Fonte: Elaborado pelo autor (2022).

Podemos identificar no retorno do comando o id da transação (1), o conteúdo da transação criptografado que é chamado de payload (2), o número do bloco (3), o bloco anterior (4), que neste caso é o bloco gênesis (lembre-se que a Blockchain armazena os registros em cadeia, ligando os registros uns aos outros, apontando quem é o bloco anterior) e por fim a assinatura do bloco, que nada mais é do que a identificação do bloco, a qual será marcada em um novo bloco como “previous_block_id”.

Executando o mesmo comando para o bloco 2 (figura 54), ou seja, a segundo voto registrado block_id:
71d6ce7de24c150bef9c8cb8ed0c27657a6ce86229e66c2c872bc2695181b7ba777f3
17bc47ffbc422476013179fb21430178b04ce94151ff91e94af89ebc8be

Figura 54: Detalhes do bloco 2

Fonte: Elaborado pelo autor (2022).

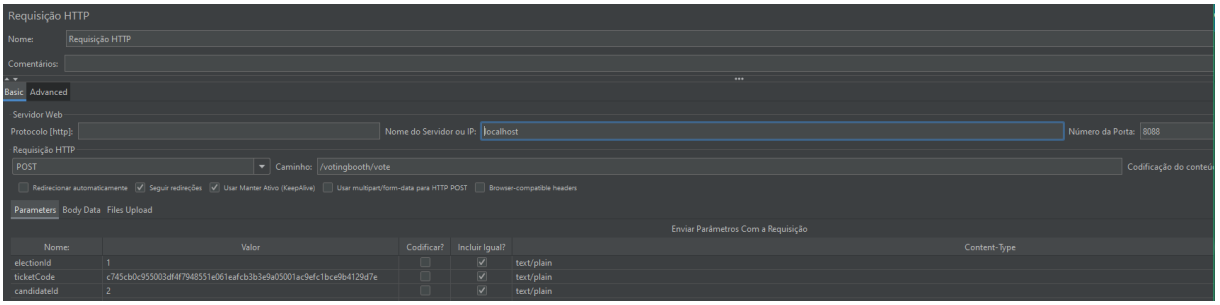
Podemos ver que as informações que identificam a transação (1) e o bloco (5) foram diferentes da anterior, mas isto é evidente, pois para cada novo bloco, novas informações são geradas. Podemos identificar que o payload (conteúdo) (2) mudou, e podemos conferir o hash do bloco anterior (4).

3.6.7 Testando a solução em modo de teste de massa

Para enviar uma massa maior de votos de forma automática, iremos utilizar a ferramenta JMeter, de código aberto e gratuita disponibilizada pela *Fundação Apache (https://jmeter.apache.org/).*

A configuração do JMeter consiste em fornecermos informações como *Nome do Servidor ou IP, Caminho da requisição, tipo de requisição e Parâmetros da solicitação,* conforme pode-se observar na figura 55:

Figura 55: Configurações da requisição HTTP através do JMeter



Fonte: Elaborado pelo autor (2022).

Realizamos várias cargas de testes, como diferentes quantidades de votos, medindo o tempo de execução do voto, o tempo mínimo e máximo da massa de execução. Na tabela 1 observamos os dados coletados:

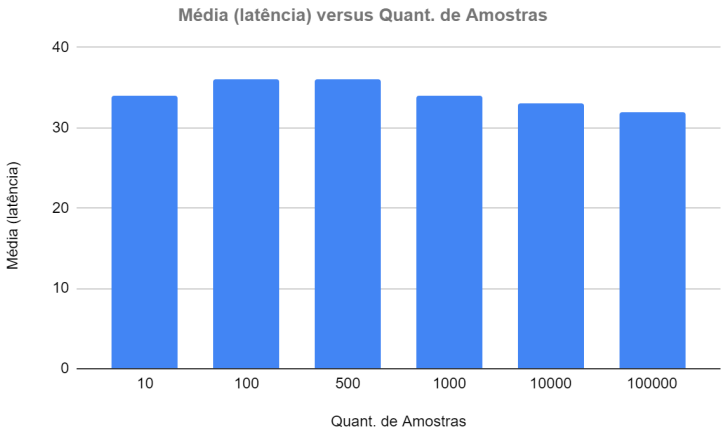
Tabela 1: Dados coletados na execução da massa de teste.

Tipo de Requisição	Quant. de Amostras (votos)	Média da latência (ms)	Lat. Mínima (ms)	Lat. Máxima (ms)
POST	10	34	26	49
POST	100	36	26	52
POST	500	36	26	187
POST	1000	34	25	265
POST	10000	33	23	261
POST	100000	32	21	627

Fonte: Elaborado pelo autor (2022)

Na figura 56 observamos o desempenho geral das cargas de testes em gráfico de colunas:

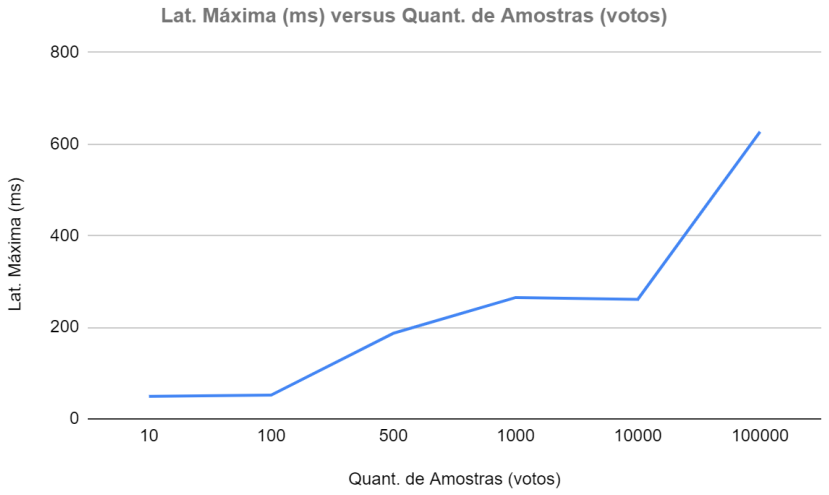
Figura 56: Desempenho médio das massas de teste.



Fonte: Elaborado pelo autor (2022)

Na figura 57 podemos observar a relação entre Quantidade de amostras e latência máxima:

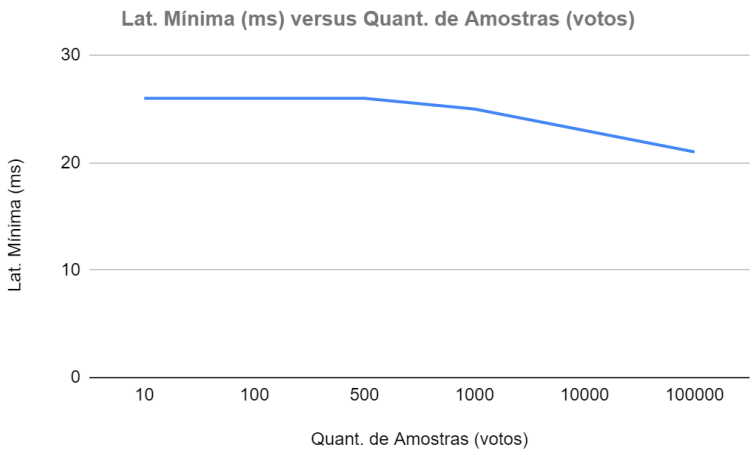
Figura 57: Relação entre Quantidade de amostras e Latência Máxima



Fonte: Elaborado pelo autor (2022)

Na figura 58 representamos a relação entre Quantidade de amostras e Latência Mínima:

Figura 58: Relação entre Quantidade de Amostras e Latência Mínima



Fonte: Elaborado pelo autor (2022)

Na seção a seguir iremos discutir os dados coletados nesta pesquisa.

4 RESULTADOS E DISCUSSÕES

Neste trabalho de conclusão de curso foi realizada a implementação de um sistema de votação seguindo os parâmetros para um problema selecionado. Para realização deste experimento foi implementado uma arquitetura baseada nos requisitos apresentados, no intuito de verificar todo o processo de votação utilizando uma Blockchain.

O experimento simula um sistema de votação generalizado, podendo ser aplicado em diversas áreas, pois conta com características comuns a um processo eleitoral, como candidatos, eleitores, momento de votação, autenticação do eleitor via chave única (simulando o documento pessoal do eleitor) e um sistema para registrar os votos.

A solução apresentada como resolução do problema de votação é baseada nos atuais processos e tecnologias, levando em consideração padrões de projetos eficientes para implementar o arcabouço necessário à aplicação. A presente proposta utilizou-se de uma Blockchain privada/permissionada, onde os usuários precisam de um acesso e aprovação para utilizar do sistema de votação.

Utilizamos do Consenso Devmode, que é um consenso personalizado para contêineres de um único nó. Para aumentarmos o número de nós de confirmação na rede, precisariamos modificar o algoritmo de consenso, a estrutura de contêineres e a especificação da máquina que o Docker está executando.

O Hyperledger Sawtooth foi utilizado como Blockchain, demonstrou-se estável e seguro para a carga de trabalho executada. Para uma carga de votações maior, se faz necessário mais testes. Uma das vantagens de utilizar o Sawtooth é que podemos conectar aplicações como módulos na Blockchain, ou seja, utilizando da Api Rest disponibilizada, é possível realizar mais transações e desenvolver contratos inteligentes mais robustos externos ao sistema de blocos.

O Docker facilitou a virtualização de uma Blockchain, sem preocupações com configuração, o que nos permitiu utilizar dos serviços Blockchain em poucos minutos. O Docker Compose (arquivo .yaml) deu agilidade na implementação da Blockchain, pois graças a comunidade do Sawtooth, com apenas um comando foi possível ter uma estrutura completa e configurada de Blockchain, caso precisássemos implementar utilizando uma linguagem de programação, levaríamos dias ou até semanas para deixar a Blockchain pronta para utilização.

Através do teste unitário na seção 3.6.6 podemos perceber que uma vantagem do sistema proposto é a segurança dos dados, pois para cada novo voto um novo bloco é criado, não permitindo que os dados sejam modificados posteriormente. Vimos através do teste que novos votos geram novas transações, etiquetando cada bloco e criando uma cadeia de blocos. Outra característica importante é que uma vez que o dado é registrado na Blockchain, não será possível removê-lo ou alterá-lo.

Já no teste de massa da seção 3.6.7 podemos observar que para 10 requisições do tipo POST (envio de 10 votos), a latência média é de 34ms, com tempo mínimo de 26ms e máximo de 40ms. Para 100 requisições do tipo POST, a latência média é de 34ms, com mínima de 26ms e máxima de 49ms. Para 500 requisições do tipo POST, a média é de 36ms, com mínima de 26ms e 187ms de máxima. Aumentando para 1000 transações, a latência média é de 34ms, com mínima de 25ms e máxima de 265ms. Aumentando em 10x a quantidade de requisições, ou seja 10.000 requisições, a média é de 33ms, com mínima de 23 e máxima de 261ms. Finalizando com 100.000 requisições, a latência média foi de 32ms, com mínima de 21ms e máxima de 627ms. Podemos observar na figura 93 onde resgatamos a relação entre quantidade de votos e latência máxima, que a medida que a quantidade de votos aumenta, o tempo de espera para processamento de algumas solicitações também aumenta, no entanto, pode-se constatar que a média de latência permanece entre 32ms e 36ms, que para os parâmetros testados e o sistema escolhido, o desempenho demonstra-se estável e escalável para uma quantidade maior de votos. Na figura 94 observamos a relação entre quantidade de requisições e latência mínima, onde os dados mostraram que a menor latência registrada é de 22ms.

Dado esses fatos, a solução proposta demonstra-se pertinente para o uso da Blockchain para um sistema de eleição, visto que há semelhanças no sistema de votação e nas características que moldam uma Blockchain: segurança, confiabilidade, escalabilidade e transparência.

Na implementação da solução abordada não codificamos regras complexas para o contrato inteligente, mas nada nos impede de criar regras e restrições para cada transação na Blockchain. Outra vantagem do Blockchain escolhido é que a arquitetura nos permite configurar um ou mais contratos inteligentes independentes

da estrutura de implementação do Sawtooth, tais contratos não irão afetar o funcionamento da Blockchain.

5 CONCLUSÃO

Neste trabalho, foram apresentados casos de usos reais relacionados à utilização da tecnologia Blockchain. Selecionamos um problema para implementar toda a teoria proposta por uma Blockchain, desde sua concepção de segurança, confiabilidade e transparência, até os registros em cadeia dos votos realizados.

Foram encontradas algumas dificuldades durante a pesquisa e implementação da solução, pois ainda existe uma escassez de materiais técnicos publicados na língua portuguesa, e neste sentido foi necessário recorrer a literaturas e artigos estrangeiros. Outro fato constatado é que evidenciamos que este é o primeiro trabalho de pesquisa técnica a ser registrado na Biblioteca da Universidade Estadual da Paraíba relacionado à implementação de uma Blockchain, ao qual nos trás mais esta responsabilidade em pesquisar e entregar um trabalho de qualidade.

Para resolver os requisitos solicitados para a solução, após bastante pesquisa foi implementado um sistema que envia requisições via Api Rest e registra em uma Blockchain permissionada. Através dos testes realizados podemos comprovar a criação dos blocos imutáveis e a identificação de cada voto realizado, demonstrando que o sistema de Blockchain não se limita somente à utilização em sistemas financeiros e criptomoedas.

De acordo com os experimentos realizados, podemos perceber que a solução apresentada atende aos requisitos de um sistema de votação, podendo ser aplicado em um caso real, como por exemplo nas eleições para reitoria de uma universidade. No entanto, novos experimentos e ampliações na rede de nós precisam ser executados e testados para garantir eficiência e mais segurança com as transações.

Por fim, podemos concluir que a tecnologia Blockchain se mostra eficiente para sistemas de votação e que limites ainda não podem ser definidos, pois a tecnologia está em constante evolução e a cada momento surgem novos estudos e investimentos empregados na melhoria da tecnologia, desta forma, na prática comprovamos que a tecnologia pode gerar promissoras respostas aos problemas atuais de confiança nos sistemas de votação.

Quanto aos trabalhos futuros, faz-se necessário mais testes para outros cenários de votação, incluindo mais nós na rede para validar o tempo que levará para as transações serem confirmadas e propagadas na rede. Também sugerimos a criação de contratos inteligentes mais complexos, eliminando possibilidades de

eleitores fraudulentos. Indo além, como proposta de utilização por parte de eleitores, pode-se ampliar a validação da aplicação realizando um teste de campo, com eleitores reais utilizando o sistema de votação e fornecendo feedback do processo através da própria plataforma.

REFERÊNCIAS

NAKAMOTO, Satoshi. **Bitcoin: A Peer-to-Peer Electronic Cash System.** 2008. Disponível em: <https://bitcoin.org/bitcoin.pdf>. Acesso em: Junho de 2022.

LAURENCE, Tiana. **Blockchain For Dummies.** New Jersey: John Wiley & Sons, Inc.,2017.

REVOREDO, Tatiana. **Blockchain: tudo o que você precisa saber.** Amazon Digital Services LLC - KDP Print US, 2019.

BASHIR, Imran. **Mastering Blockchain : a deep dive into distributed ledgers, consensus protocols, smart contracts, DApps, cryptocurrencies, Ethereum, and more**. Packt Publishing; 3ª edição, 2020. Disponível: <https://www.amazon.com.br/Mastering-Blockchain-distributed-consensus-cryptocurrencies-ebook/dp/B08GMCXNFV>. Acesso em: Julho de 2022.

XU, Xiwei; WEBER, Ingo; STAPLES, Mark. **Architecture for Blockchain Applications.** Springer, 2019.

HYPERLEDGER**. An Introduction to Hyperledger.** Hyperledger Foundation, 2018. Disponível em: <https://www.hyperledger.org/wp-content/uploads/2018/08/HL_Whitepaper_IntroductiontoHyperledger.pdf> Acesso em Junho de 2022.

HYPERLEDGER. **Hyperledger Architecture, Volume 1.** Hyperledger Foundation, 2017**.** Disponível em:<https://www.hyperledger.org/wp-content/uploads/2017/08/Hyperledger_Arch_WG_Paper_1_Consensus.pdf>. Acesso em Junho de 2022.

SILVEIRA, Carlos Marcelo da. **Do voto em papel ao eletrônico: estudo de caso da implantação do voto biométrico em Canoas/RS, 2011**. Disponível em https://repositorio.ufsm.br/handle/1/425. Acesso em Junho de 2022.

APARECIDA, Roseli. **A proteção dos dados pessoais face às novas tecnologias.** São Paulo. Disponível em: <http://www.publicadireito.com.br/artigos/?cod=d1aae872c07c10af>. Acesso em Junho de 2022.

SINGHAL, Bikramaditya; DHAMEJA, Gautam; SEKHAR, Priyansu. **Beginning Blockchain: A Beginner 's Guide to Building Blockchain Solutions**. Apress, 2018.

EJEKE, Patrick. **WEB3: What Is Web3? Potential of Web 3.0.** 2021. Disponível em: <https://www.amazon.com/gp/product/B09TZFDS4R/ref=dbs_a_def_rwt_bibl_vppi_i0> Acesso em Junho de 2022.

SALES, Guilherme. **Ascensão das Criptomoedas – oportunidades, impactos e riscos.** Peers, 2021. Disponível em: <https://peers.com.br/ascensao-das-criptomoedas/> Acesso em: Julho de 2022.

Forbes Top 50 Blockchain: conheça as empresas que usam a tecnologia. Forbes, 2022. Disponível em: <https://forbes.com.br/forbes-money/2022/02/forbes-top-50-Blockchain-conheca-as-empresas-bilionarias-que-utilizam-a-tecnologia/> Acesso: Junho 2022.

UNIAS, Antônio. **Estudo de arquiteturas dos Blockchains de Bitcoin e Ethereum.** Unicamp, 2016. Disponível em: <https://www.fee.unicamp.br/sites/default/files/departamentos/dca/eadca/eadcaix/artigos/lucena_henriques.pdf>. Acesso em: Junho de 2022.

FRANKENFIELD, Jake. **Merkle Root (Cryptocurrency).** Investopedia, 2021. Disponível em: <https://www.investopedia.com/terms/m/merkle-root-cryptocurrency.asp>. Acesso em: Junho de 2022.

Árvore de Merkle. Disponível em: <https://www.seekpng.com/ima/u2q8y3i1t4u2w7i1/>. Acesso em: Junho de 2022.

FERRAZ, Lucas. **Criptografar MD5, SHA1 e SHA256**. Lucas Ferraz. Disponível em: <https://lucasferraz.com.br/ferramentas/criptografar-md5-sha1-sha256/>. Acesso em: Junho de 2022.

O que é SHA-256?. Bit2me Academy. Disponível em: <https://academy.bit2me.com/pt/sha256-algoritmo-bitcoin/>. Acesso em Junho de 2022.

HOINASKI, Fábio. **Tipos de Blockchain: Qual o melhor para a cadeia de suprimentos?.** Blog Ibid System Solutions, 2021. Disponível em: <https://www.ibid.com.br/blog/tipos-de-Blockchain-qual-o-melhor-para-a-cadeia-de-suprimentos/>. Acesso em: Junho de 2022.

AKKOYUNLU, E.; EKANADHAM, K.; HUBER, R. **Some constraints and trade-offs in the design of network communications.** Portal ACM Digital Library, 1975. Disponível em:<https://dl.acm.org/doi/10.1145/800213.806523>. Acesso em: Junho de 2022.

Building Blocks: Blockchain network for humanitarian assistance - Graduated Project. Portal WFP, 2022. Disponível em: **<**http://www.wfp.org https://innovation.wfp.org/project/building-blocks>. Acesso em: Julho de 2022.

Projeto Federal Conecte SUS. Conecte SUS, 2022. Disponível em:<https://conectesus.saude.gov.br/home>. Acesso: Junho de 2022.

Estratégia de Saúde Digital para o Brasil 2020-2028. Portal Saúde GOV. Disponível em: <https://bvsms.saude.gov.br/bvs/publicacoes/estrategia_saude_digital_Brasil.pdf>. Acesso em: Junho de 2022.

AGUIAR, Letícia. **Fake News em tempo de eleições**. Repositório Institucional AEE, 2020. Disponível em: <http://repositorio.aee.edu.br/bitstream/aee/16880/1/Monografia%20-%20LETICIA%20AGUIAR.pdf>. Acesso em: Junho de 2022.

Using HTTP Methods for RESTful Services. Rest Api Tutorial, 2022. Disponível em:<https://www.restapitutorial.com/lessons/httpmethods.html#:~:text=The%20primary%20or%20most%2Dcommonly,but%20are%20utilized%20less%20frequently>. Acesso em: Junho de 2022.

Setting Up a Sawtooth Node for Testing. Hyperledger Sawtooth, 2022. Disponível em:<https://sawtooth.hyperledger.org/docs/1.2/app_developers_guide/installing_sawtooth.html#docker>. Acesso em: Junho de 2022.

Documentação Docker. Disponível em:<https://docs.docker.com/>. Acesso em: Junho de 2022.

GAMMA, Erich et al. **Padrões de projeto: soluções reutilizáveis de software orientado a objetos**. Porto Alegre : Bookman, 2007.

DAO Pattern: Persistência de Dados utilizando o padrão DAO. Portal Devmedia, 2022. Disponível em: <https://www.devmedia.com.br/dao-pattern-persistencia-de-dados-utilizando-o-padrao-dao/30999#:~:text=Classes%20DAO%20s%C3%A3o%20respons%C3%A1veis%20por,em%20instru%C3%A7%C3%B5es%20SQL%20e%20mandar>. Acesso em: Junho de 2022.

SOMMERVILLE, Ian. **Engenharia de Software**. São Paulo: Addison Wesley, 2003.

REZENDE, Denis A. **Sistemas de Informações Organizacionais: guia prático para projetos em cursos de administração, contabilidade e informática.** São Paulo: Atlas, 2005.

Apêndice no TCC: o que é, exemplo e como colocar no trabalho. Regras para TCC, 2019. Disponível em: <https://regrasparatcc.com.br/elementos/apendice-no-tcc/#:~:text=Na%20hora%20de%20incluir%20ap%C3%AAndices,formata%C3%A7%C3%A3o%20do%20restante%20do%20relat%C3%B3rio.&text=Caso%20se%20esgotem%20as%2023,ABNT%20recomenda%20usar%20letras%20dobradas.>. Acesso em: Julho de 2022.

APÊNDICE A - Código-fonte da Solução

Aplicação disponível no Github: https://github.com/jonathantecsantos/sistema_votacao

APÊNDICE B - Código-fonte da Implementação Front-End

Logo abaixo a implementação de algumas abstrações do front-end:

Figura 59: Arquivo de tratamento do contrato inteligente

```
'use strict'

const { TransactionProcessor } = require('sawtooth-sdk/processor');
const VALIDATOR_URL = 'tcp://localhost:4004';

module.exports = (handler) => {
  const tp = new TransactionProcessor(VALIDATOR_URL)
  tp.addHandler(handler)
  tp.start()
}
```

Fonte: Elaborado pelo autor (2022).

Implementação do arquivo ***voteHandler.js***:

Figura 60: Arquivo do contrato inteligente - voteHandler.js

```
'use strict'

const { createHash } = require('crypto')
const { TransactionHandler } = require('sawtooth-sdk/processor/handler')
const {InvalidTransaction} = require('sawtooth-sdk/processor/exceptions');
const {Decoder} = require('cbor')
const {calculateVoteAddress,handlerInfo} = require('./infra');

// Encoding helpers and constants
const getAddress = (key, length : number  = 64) => {
  return createHash( algorithm: 'sha512').update(key).digest( encoding: 'hex').slice(0, length)
}

const encode = obj => Buffer.from(JSON.stringify(obj, Object.keys(obj).sort()))

class VoteHandler extends TransactionHandler {
  constructor () {
    console.log('Iniciando smart contract para votos ')
    const info = handlerInfo();
    super(info.family,  versions: [info.version],  namespaces: [info.prefix]);
  }
  apply (txn : TpProcessRequest , context : Context ) {
    console.log("chegando uma nova transacao");

    const dataDecoded = Decoder.decodeFirstSync(txn.payload);
    const payload = JSON.parse(dataDecoded);

    const blockAddress = calculateVoteAddress(payload)
    const {candidateNumber,ellectionName} = payload;

    return context.setState( addressValuePairs: {
      [blockAddress]: encode( obj: {candidateNumber,ellectionName})
    });
  }
}

module.exports = {
  VoteHandler
}
```

Fonte: Elaborado pelo autor (2022).

Implementação do arquivo infra.js:

Figura 61: Arquivo de configuração do sdk sawtooth - infra.js

```
const {createHash} = require('crypto')
const {protobuf} = require('sawtooth-sdk')
const {createContext, CryptoFactory} = require('sawtooth-sdk/signing')
const {Secp256k1PrivateKey} = require('sawtooth-sdk/signing/secp256k1')
const request = require('request');

function handlerInfo(){
  const familyName = 'onlinevoting';
  return {
            prefix : getAddress(familyName,  length: 6),
            family : familyName,
            version :'0.0.1'
        };
}

function getAddress(key, length) {
  return createHash( algorithm: 'sha512').update(key).digest( encoding: 'hex').slice(0, length)
}

function calculateVoteAddress(payload) {
  return handlerInfo().prefix + getAddress(payload.ellectionName, length: 20) + getAddress(payload.userNumber, length: 20)
      + getAddress(payload.address, length: 24);
}

const getAssetAddress = payload => handlerInfo().prefix + getAddress(payload.ellectionName, length: 20)
    + getAddress(payload.userNumber, length: 20) + getAddress(payload.address, length: 24)

const getAssetAddress = payload => handlerInfo().prefix + getAddress(payload.ellectionName, length: 20)
    + getAddress(payload.userNumber, length: 20) + getAddress(payload.address, length: 24)

function sendToSawtoothApi(batchBytes) {
  request( uri: {
      url: 'http://localhost:8008/batches?wait',
      method: 'POST',
      body: batchBytes,
      encoding: null,
      headers: {'Content-Type': 'application/octet-stream'}
    },  options: (error, response, body) => {
      if (error) {
        console.log(error);
      } else {
        const res = new Buffer(response.body,  encoding: 'base64').toString()
        console.log('Response: ', res);
      }
    })
```

Fonte: Elaborado pelo autor (2022).